

**Application of Mathematical Methods in the Study of Chemical Dynamical  
Systems to an Inventory Multiplier-Accelerator Business Cycle Model**

by  
Benjamin Bui

Professor Peacock-López, Advisor

A thesis submitted in partial fulfillment  
of the requirements for the  
Degree of Bachelor of Arts with Honors  
in Chemistry

Williams College  
Williamstown, Massachusetts

May 1, 2019

## Acknowledgements

I would like to express my gratitude to my advisor, Professor Enrique Peacock-López for supporting me throughout my thesis. It was always helpful to have someone I could bounce ideas off of and to explain to me obtuse math I would find in the literature. You really let me develop and understand the material needed independently but were always there when I needed help. You were the first professor I did research with in the summer of my freshman year and it was a pleasure to work on my thesis with you in my senior year.

I would also like to thank my suite and the people of the Jesup community for providing me with both the emotional and technical support throughout my thesis. You provided the breaks after long hours of work and I look forward to staying close to all of you long after I graduate

Thank you to the faculty and students Chemistry Department of Williams College for being with me all of these years. I understand that my thesis appears unorthodox for the department but I thank you all for being understanding and willing to let me work on this. Special thanks to Brandon Vuong, Si Hou Lon, and Alison Wong for making sure I understood what I could expect the average chemistry student to know and understand.

I of course must mention my family who were instrumental in raising me and helping reach the point I am now. Thank you to my mother and father, to my brother and sister: Helen and Bryant, and to the rest of my extended family to whom I love dearly.

## **Abstract**

The field of chemistry and physics has made many contributions to economic theory in the past. The development of the theory of Brownian motion and other stochastic processes proved revolutionary in economics and finance with an entire subfield of econophysics being developed in order to accommodate these new ideas. Many notable processes in chemistry and physics are represented as chaotic dynamical systems; however, chaos is still not a commonplace point of discussion in macroeconomic theory. In this paper, I present two previously created business cycle models, a multiplier-accelerator model with a Robertson lag by Tönu Puu and an inventory cycle model with a Lundberg lag by Metzler and Wegener, before creating a new model that incorporates these elements from these two models. This model explains the inability for firms to accurately determine future outcomes on the potentially chaotic or quasi-periodic behavior of output levels. I find that even given reasonable parameter choice, it is reasonable for the model to be in a chaotic regime, thus allowing business cycles to be driven endogenously as opposed to through external, stochastic shocks.

# Contents

<b>1</b>	<b>Introduction</b>	<b>5</b>
1.1	Background on Dynamical Systems . . . . .	5
<b>2</b>	<b>Multiplier-Accelerator Business Cycles</b>	<b>14</b>
2.1	Background . . . . .	14
2.2	Model Set-up . . . . .	15
2.3	Growth Dynamics . . . . .	17
<b>3</b>	<b>Heterogenous Inventory Cycles by Wegener et al.</b>	<b>22</b>
3.1	Background . . . . .	22
3.2	Model Set-Up . . . . .	22
3.3	Analysis of Income Dynamics . . . . .	24
<b>4</b>	<b>Inventory Cycles with Averaged Expectations, Endogenous Investment, and Consumption Lag</b>	<b>29</b>
4.1	Model Set-Up . . . . .	29
4.2	Growth Dynamics . . . . .	33
4.3	Chaotic Behavior in Growth . . . . .	41
<b>5</b>	<b>Conclusion</b>	<b>49</b>
<b>6</b>	<b>References</b>	<b>55</b>
<b>A</b>	<b>Solving for Change in Income</b>	<b>57</b>
<b>B</b>	<b>Computational Analysis of the Logistic Map</b>	<b>60</b>
B.1	Cobweb Plot . . . . .	60
B.2	Bifurcation Diagram . . . . .	61
B.3	Lyapunov Plot . . . . .	62

<b>C Computational Analysis of the Multiplier-Accelerator Model</b>	<b>65</b>
C.1 Cobweb Plot . . . . .	65
C.2 Bifurcation Diagram . . . . .	67
C.3 Lyapunov Plot . . . . .	68
<b>D Computational Analysis of the Metzlerian Inventory Cycle Model with Heterogenous Expectation Rules</b>	<b>70</b>
D.1 Timeseries Plot . . . . .	70
D.2 Cobweb Plot . . . . .	72
D.3 Bifurcation Diagram . . . . .	74
<b>E Computational Analysis of the Metzlerian Inventory Cycle Model with Averaged Expectations, Endogenous Investment, and Consumption Lag</b>	<b>80</b>
E.1 Timeseries Plot . . . . .	80
E.2 Investment Curves . . . . .	81
E.3 Bifurcation Diagram Varying Parameters . . . . .	83
E.4 Bifurcation Diagram Varying Initial Conditions . . . . .	87
E.5 Bifurcation Analyzer . . . . .	92
E.6 Results of Bifurcation Analysis . . . . .	104
E.7 Lyapunov Plot varying Parameters . . . . .	109

# 1. Introduction

The study of dynamical systems is traditionally thought to have begun with the publication of "New Methods of Celestial Mechanics" by Poincaré and expanded with the work of Lyapunov into a theory of the stability of dynamical systems. It was not until the 1960s however that the use of chaos and stability theory exploded across disciplines.<sup>1</sup>

Dynamics are typically an unnecessary tool when studying the paths of chemical reactions. Their value became apparent however, when Belousov and later Zhabotinsky released published their work on an oscillatory reaction, a reaction that would later come to be known as the BZ reaction.<sup>2</sup> Cycles were also known to exist in the biochemical realm with many famous pathways in organisms such as the Krebs cycle and Calvin cycle; however the BZ reaction was developed to create an inorganic analogue to the Krebs cycle.<sup>3</sup> The development of this cycle allowed for a relatively easily replicable cyclic reaction with with easily measurable indicators of the progress of the reaction.

The study of dynamical chemical systems has since expanded to a variety of other mechanisms such as self-replicating molecules<sup>4</sup> in addition to studying fractal patterns and dimensionality involved in electrochemical deposits<sup>5</sup> and flame patterns.<sup>6</sup> Despite the fairly wide range of background information required to set up these different models, the underlying mathematical theory used to study these models is identical which has contributed to the wide range of interdisciplinary work performed by theorists in the field.

## 1.1 Background on Dynamical Systems

Traditional dynamical systems are modelled in continuous time as a system of ordinary differential equations. These systems typically treat time as the singular independent variable and solve for the evolution of one or more variables in terms of time. A classic continuous time dynamical system is the simple pendulum which allows one to model the movement of a pendulum in space in terms of time. This model uses a variety of simplifying assumptions in order to reduce the problem of the pendulum into a single variable function, holding the length of the pendulum and acceleration due to gravity

constant.

$$\frac{d^2}{dt^2} + \frac{g}{L} \sin \theta = 0 \quad (1.1)$$

Most attempts at modelling real-world systems however require the use of multiple dependent variables. However, as the amount of variables increases, the complexity of the model increases. Modelling an  $n$ -body system acting on each other gravitationally is of obvious interest in astronomy; however, it was quickly found that although a 1 and 2 body system were relatively simple to solve for, the introduction of a third body caused significant complications. The 3-body system is in fact what Poincaré studied in order to develop a theory of chaotic deterministic systems.<sup>7</sup>

That is not to say that these systems cannot arrive at ordered solutions. Although continuous time systems require a minimum of 3 variables in order for chaos to arise, there are typically windows of order in chaotic regimes that allow for stable, oscillatory behavior. The BZ-reaction is still being actively studied and is known to actually be highly complex but is often reduced into 7 primary sub-reactions.<sup>3</sup> A great deal of the work involved in studying dynamical systems is actually on finding ways to simplify models in order to arrive at more mathematically tractable systems. The BZ-reaction has been simplified to a 3-variable system that still provides the complex periodicity and chaotic behavior characteristic of the model.<sup>8</sup>

Although many processes in the real-world are more intuitively interpreted as operating as a continuous time function, there are many occasions where it is possible and in fact beneficial to think in a discrete-time sense. The prevalence of this type of system varies depending on the exact field of study; however, it is important to note that when computationally modelling continuous time systems, it is impossible for computers to truly operate with continuous variables and thus even these systems are reduced to technically discrete models.

Population dynamics are frequently analyzed as a discrete time system as opposed to continuous time. It is often of more practical use to interpret  $t = 0, 1, 2, 3, \dots$  as the change in population per year or per season as opposed to determining the change in population over infinitesimally small changes in time, especially when considering smaller population sizes. In terms of technique, many of the mathematical principles used in analyzing dynamic systems in continuous time apply to discrete time systems; however, it would be a mistake to assume the two were identical. An important distinction between the two is the nature by which chaos can occur. As described previously, a continuous time system requires 3 variables in order for chaos to occur, sometimes referred to as a 3-dimensional phase space. A discrete time system only requires 1 variable in order to display the same type of chaotic behavior, i.e. discrete time systems can be of arbitrary

dimension for chaos to arise.

The systems discussed throughout this paper will be of the discrete variety because of common convention and ease of interpretability. Laws pertaining to the physical world are, for all intents and purposes, scalable to the infinitesimal degree which allows for their use in continuous models. Economic models do not have such a basis in physical laws. It is also important to note that, due to the complexity involved in the human behavior that economic models are trying predict, the exact numerical values of such a high-level model are typically of minor concern. The general behavior of the model is significantly more valuable in order to determine the effects of an economic assumption.

The logistic map is regarded as the prototypical chaotic discrete time mapping. The logistic function, which the logistic map is based off of, was developed to study population dynamics but actually garnered widespread use in other disciplines such as the study of autocatalytic reactions, computer science, statistics, and economics.<sup>9</sup>

$$\frac{d}{dx}f(x) = f(x)(1 - f(x)) \quad (1.2)$$

The logistic function has 2 equilibria or points where the derivative of the function is 0.  $f(x) = 0$  is an unstable equilibrium but  $f(x) = 1$  is a stable equilibrium point which means that other points on the function will tend towards this equilibrium overtime. This can be realized by solving for the derivative of the function at points when  $f(x) \in (0, 1)$  which is universally positive and  $f(x) \in (1, \infty)$  which is negative. Integrating the differential equation gives the general form equation:

$$f(x) = \frac{e^x}{e^x + C} \quad (1.3)$$

This function gained prominence due to its rapid, exponential growth when  $f(x)$  is low and its slow, linear decaying to non-existent growth as population increases. Used by notable mathematicians operating in the field of population dynamics such as Verhulst, Pearl, and Lotka, the model continues to be widely used today and is often the basis upon which other modifications are applied.<sup>10</sup>

$$x_{t+1} = \mu x_t(1 - x_t) \quad (1.4)$$

The logistic map is a difference equation model popularized by Robert May as a discrete time analogue to the logistic function.<sup>11</sup> When interpreted in the biological context,  $x_t$  refers to the ratio of the population at time  $t$  compared to the maximal population, thus the mapping is bounded between 0 and 1. Here we see the conceptual difference between

discrete and continuous time systems. Differential equations solve for the derivative of a variable with respect to time in terms of the variable; difference equations solve for the actual state of the variable in the successive state provided we know the state of the variable in the previous time period. Much like how differential equations can be of higher order with the introduction of higher order derivatives, a difference equation can also be of higher order by including more time period of lag in the function for the state of the variable which is valuable in a variety of the models discussed later.

Much like how the equilibrium points were solved for in the differential equation, difference equations also have equilibrium points where  $x_{t+1} = x_t$ . Interestingly, unlike the logistic function, there does not exist a fixed point at  $x_t = 1$  as this would result in  $x_{t+1} = 0$ . Solving for the fixed points, we have:

$$x_{t+1} = x_t = 0, \frac{\mu - 1}{\mu} \quad (1.5)$$

The stability of a fixed point is again dependent on the derivative of the function; however, there are differences in the details of our analysis. Treating  $f(x) = x_{t+1}$ , we see the derivative of the logistic map is:

$$f'(x) = \mu(1 - 2x) \quad (1.6)$$

For reasons that will become clear later, the stability of a point on the map requires that  $|f'(x)| < 1$ . Solving for when this is true for our two fixed points, we see that  $\mu < 1$  provides stability for the origin fixed point and  $1 < \mu < 3$  gives stability for the non-zero fixed point. Thus, provided the parameter  $\mu$  satisfies either of the conditions set previously, it will converge to one of the fixed points in a relatively small, finite amount of iterations.

This behavior can be visualized using a cobweb diagram. This diagram consists of 3 primary elements: a plot of the mapping, a  $45^\circ$  line, and a plot of the variable's trajectory. An example of a cobweb diagram can be seen in Figure 1.1. This diagram shows the trajectory of  $x$  starting at a value of 0.1 when there is a stable, non-trivial fixed point.

The  $45^\circ$  line is defined as the line where  $x_t = x_{t+1}$  which is useful for determining the result of successive iterations. Beginning from the point  $x_0$ , we can then determine what point  $x_1$  will be via the mapping. We can then look horizontally to the  $45^\circ$  line until we intersect with it. The  $x$ -coordinate of this intersection point is equivalent to the result of the mapping of the previous iteration, thus using this new point will allow us to determine the result of the next iteration of the function. This process can be repeated

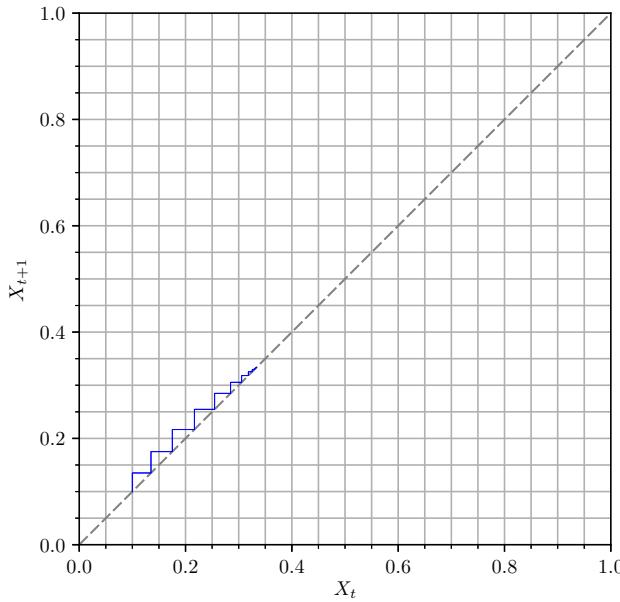


Figure 1.1: Cobweb plot of the logistic map setting  $\mu = 1.5$  and  $x_0 = 0.1$ . The trajectory asymptotically approaches the equilibrium point of  $\frac{1}{3}$ .

ad infinitum; however, the result will soon prove uninteresting for stable points and orbits as the trajectory will converge and repeat its behavior.

The reason the logistic map is so frequently studied is because of its ability to exhibit complex behavior beyond a stable equilibrium solution. Once  $\mu > 3$ , the mapping enters a cyclic region. Much like how fixed points could be solved for by identifying where  $x_t = x_{t+1}$ , stable oscillatory points can be found by solving for the equilibrium points of higher iterations of the function. A 2-cycle will be such that  $x_t = x_{t+2} \neq x_{t+1}$  for example and the stability of a such a cycle can be found using the same methodology as described previously. The logistic map also provides a mechanism to more quantitatively describe what it means for a system to be chaotic. The Lyapunov exponent, named after one of major driving forces in the development of stability analysis, is used to measure the effect of small perturbations in initial conditions on the trajectory of the variable.<sup>12</sup> Conceptually, the logistic map and the systems discussed in this paper are deterministic. However, chaotic systems have highly divergent trajectories with even small changes in their initial conditions; thus knowing approximately what the initial conditions are does not provide approximate information on the trajectory of the variable.

In order to quantify this in a first order system, we begin by taking the absolute value of the derivative of the function as this allows us to effectively magnify the effect

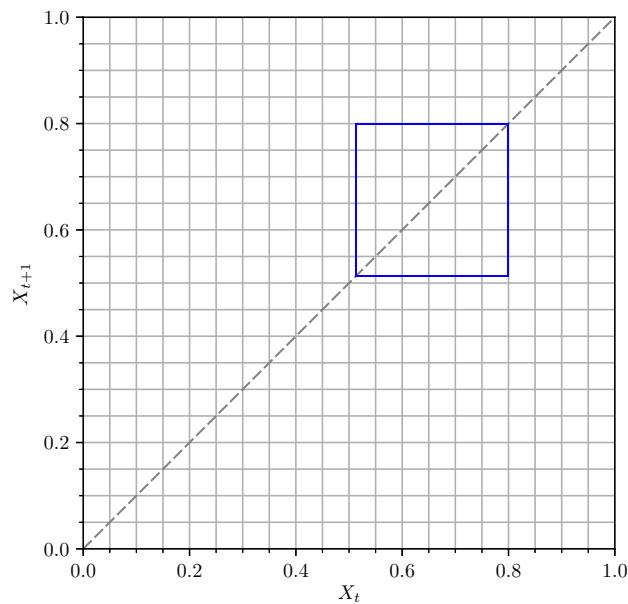


Figure 1.2: 2-period cycle of the logistic map showing only the cyclic behavior.

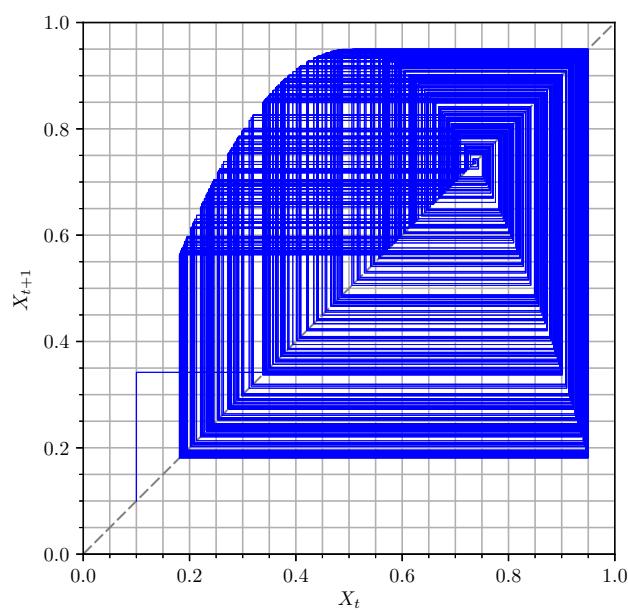


Figure 1.3: Chaotic behavior in the logistic map.

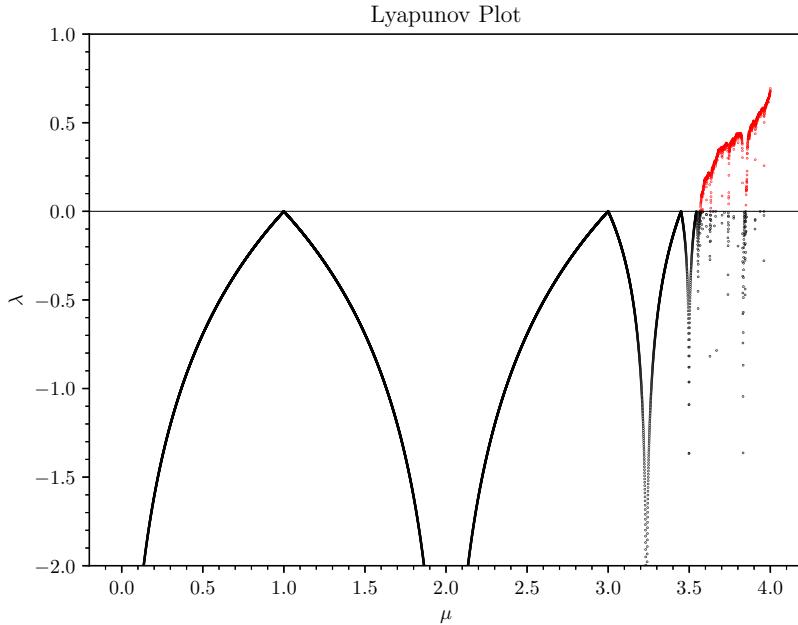


Figure 1.4: Lyapunov exponent plotted against  $\mu$  for the logistic map. Initial value of 0.1 is used. Red denotes regions where  $\lambda \geq 0$ , black denotes regions where  $\lambda < 0$ .

of an infinitesimal change in the initial conditions. We then take the natural logarithm of this derivative in order to measure the exponential rate of separation of trajectories. Finally, we take the average separation over an arbitrarily high number of iterations  $n$  as exponential separation is not necessitated over all phase space. This gives us the equation:

$$\lambda_n(x_0) = \frac{1}{n} \sum_{t=1}^{t=n} \ln|f'(x_{t-1})| \quad (1.7)$$

where  $\lambda_n(x_0)$  is the Lyapunov exponent for a given initial point when allowed to run for  $n$  iterations. The true value of the Lyapunov exponent is the limit of the infinite series as  $n \rightarrow \infty$  divided by  $n$ ; however, the complexity of these maps often makes it practically impossible to analytically solve for the limit. By choosing arbitrarily high values of  $n$  though, it is possible to achieve better approximations at the expense of computational time. It is also important to note that, although the Lyapunov exponent is a function of the initial condition, as long as the initial state is not in some stable fixed point or cycle, the trajectories will follow that of the chaotic attractor, thus the value of the Lyapunov exponent should be mostly consistent regardless of the choice of initial conditions although this assumption is less robust in the presence of multiple attractors.

Another way visually see the behavior is with a bifurcation diagram. This diagram

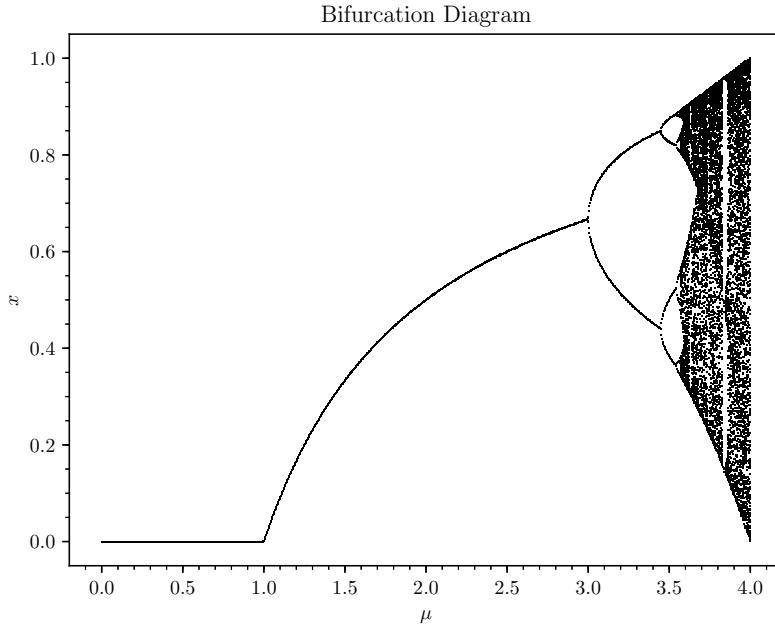


Figure 1.5: Bifurcation diagram plotting  $x$  against  $\mu$  for the logistic map.

shows the long run behavior of the variable for a given variable. Figure 1.5 qualitatively shows the behavior described previously. For parameter values between 0 and 1, we see the origin fixed point is stable. For parameter values between 1 and 3, there is still a single fixed point that is monotonically increasing; however, we can also clearly see the beginning of the 2-period cycle once the parameter exceeds 3. It is difficult however, to determine when predictable higher-order cyclic behavior ends and chaotic behavior begins via qualitative observation of the bifurcation diagram. The benefit of the bifurcation diagram is that it allows us to see both where the bifurcation points are and what behavior these points signify. Bifurcation points are where infinitesimally small quantitative changes in the parameter induce significant qualitative or topological change in the behavior of the mapping such as the transition from a stable fixed point to a 2-period cycle.

Research on the logistic map and other iterated maps has shown the existence of what is called Feigenbaum's constant. This constant can be found by observing the behavior of the periodic cycles of the map. The interval of stability decreases and the ratio of subsequence intervals actually approaches a limit  $\delta \approx 4.6692$ .<sup>12</sup> All other topologically similar maps with a single local maximum share this Feigenbaum constant. Once the mapping exceeds this constant, chaos occurs which allows for another method to determine precisely where chaotic behavior occurs.

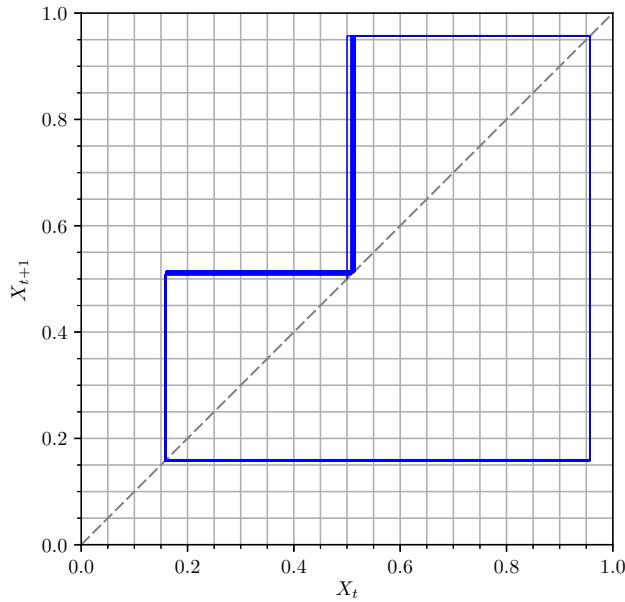


Figure 1.6: 3-period cycle of the logistic map showing only the cyclic behavior.

It is also beneficial to point out that another mechanism exists for cyclic behavior to exist. The previously described method involved taking a mapping and solving for the stability of its double iteration. This allows for  $2n$ -period stability cycles to exist. However, there does exist odd-ordered cycles such as the 3 cycle; however, it occurs as a window of order in the region of chaos. These windows can be seen in Figure 1.4 where the Lyapunov exponent dips into the negative region past the chaotic bifurcation point. These bifurcation points are known as tangent bifurcations. This also allows us to use Sharkovsky's Theorem, which states that any continuous mapping with a 3-period cycle must also have every  $n$ -period cycle for every  $n \in \mathbb{Z}$ .<sup>12</sup> A variety of other mathematical techniques exist to study the dynamics of difference equation mappings but these will be covered when necessary in the paper. Most notably, the determination of chaos in higher order models requires a slightly different but conceptually similar technique to the first order case.

## 2. Multiplier-Accelerator Business Cycles

### 2.1 Background

John Maynard Keynes work revolutionized economic thought; however, he never formalized any of his theories into a mathematical theory. This was performed in a process known as the neoclassical synthesis which was so called for its attempt to bridge classical models with these Keynesian principles. Paul Samuelson and Sir John Hicks were deeply influential in summarizing Keynesian macroeconomics into mathematically tractable models and their work was incorporated into a discrete-time business cycle model by Tönuu Puu.<sup>12</sup> Key to the theory of the model are the concept of a multiplier and accelerator.

Keynes espoused the idea that increasing spending in the economy by some amount  $x$  actually increased the national output by an amount  $> x$ . In essence, the effect of increased spending was actually multiplied by a positive amount. Although this was originally studied in order to determine the effects of changes in government spending in order to inform fiscal policy,<sup>13</sup> the principle of the mechanism impacts all forms of expenditure in the economy.

Integral to the rise of cyclic dynamics in the model is the presence of the Keynesian accelerator which actually makes use of the multiplier effect. The effect can be best explained via example: suppose there is a general increase in national output level. This increases business profits and expectations, thus inducing them to increase investments. This further increases output via the multiplier effect, causing an accelerating level of growth. The same occurs if the economy depresses as businesses will decrease investment due to falling prospects<sup>14</sup>

The combination of these two mechanisms, typically referred to as the multiplier-accelerator effect, are sufficient in creating a stable cyclic or even chaotic economy as opposed to a static steady-state model. This type of model allows for cyclic behavior to occur endogenously, that is to say the model allows for persistent behavior outside of the steady state. This idea runs contrary to the idea that economic booms and recessions are

reactions to exogenous shocks which is common in the new classical models. Also known as freshwater economics, these models assume that agents are perfectly rational and are capable of learning from past experiences. This viewpoint came into prominence in the late 1970s with a model by Lucas and Sargent<sup>15</sup> which sought to move past these seemingly outdated Keynesian principles. In the modern day however, Keynesianism has regained popularity with a new neoclassical synthesis that resulted in a new school of thought, New Keynesianism, that seeks to provide stronger micro-foundations to macroeconomic models than was previously encountered in neo-Keynesianism.

## 2.2 Model Set-up

The model presented here is identical to that presented by Tönu Puu.<sup>12</sup> The first factor of the economy to consider is that of investment. In order for investment to operate under Keynes' accelerator principle, capital stock must be in a proportion to the change in income, thus the investment level would be a function of the rate of change in income.

A linear function for investment captures this premise; however, this leads to unrealistic behavior for higher magnitudes of income change. Suppose income dramatically increased; a linear function implies that a proportionally high level of investment can sustain this higher level of production when in reality other factors of production such as the land, labor, or technology available are the primary limiting factors. Of larger concern with a linear model though is if the economy encounters a sharp decrease in income. This induces a large, negative value for investment which implies that firms would actively destroy their machinery and other forms of capital stock in the event of an economic recession. This is obviously unrealistic and so John Hicks introduced a piecewise linear investment function such that at extreme levels of income change, investment will reach a predetermined maximal or minimal value. This piecewise function was then adapted to be differentiable over all points by Richard Goodwin by approximating the curve with a hyperbolic-tangent function.<sup>12</sup>

Puu approximates the hyperbolic-tangent function with its linear-cubic Taylor series expansion as this introduces a back-bending behavior into the curve. This allows the investment curve to capture not only firm behavior in the private sector but also implicitly include government spending and taxation. This follows from the now common policy for governments to engage in contracyclic behavior, increasing the quantity and size of spending projects and decreasing taxes when income is decreasing. Likewise, when the economy is performing well, the government cuts back on spending projects intended to stimulate the economy while also increasing taxes in order to take advantage of the

overheating economy. We can thus write the function for investment as:

$$I_t = v(Y_{t-1} - Y_{t-2}) - v(Y_{t-1} - Y_{t-2})^3 \quad (2.1)$$

A limitation of this formulation is that investment behaves in an unrealistic manner if the absolute value of income change exceeds unity but this is not a problem in the model for reasons we will cover later.

Consumers are expected to adjust their consumption relative to the level of income by some marginal propensity to consume:  $1-s$  ( $s$  can be thought of as the marginal propensity for consumers to save). However, Puu incorporates a Robertson lag into the model by making current consumption a function of lagged income. Moreover, consumption in the current period includes all income saved from the 2 lagged period. Expressed as a function:

$$C_t = (1-s)Y_{t-1} + sY_{t-2} \quad (2.2)$$

This function incorporates a 1-period Lundberg lag where  $s \in [0, 1]$  is the marginal propensity to save. This function also contains a 2-period delayed consumption due to the marginal propensity to save, thus all income made in some period  $t$  can be thought of as being eventually spent in the period  $t+1$  and  $t+2$ . Although intuitive, this explanation is not wholly accurate as the Lundberg lag does not imply saving of income to spend in the next period but rather that spending behavior is influenced only on the information of lagged income level. The choice of  $s$  dictates the relative importance of lagged income, a higher marginal propensity to save reduces the impact of income made in the previous time period but increases the effective impact of the income made two time periods ago.

This simplified economy only has consumption and investment as its factors, thus

$$Y_t = I_t + C_t \quad (2.3)$$

However, as this model is unbounded, it is more useful to analyze the growth rate of the mapping as opposed to the raw output. Defining

$$\dot{Y}_{t-1} \equiv Y_t - Y_{t-1} \quad (2.4)$$

This growth rate can be solved for as

$$\dot{Y}_t = (v-s)\dot{Y}_{t-1} - v\dot{Y}_{t-1}^3$$

$s$  has a real meaning behind its value but as the value of  $v$  is dependent on the value of

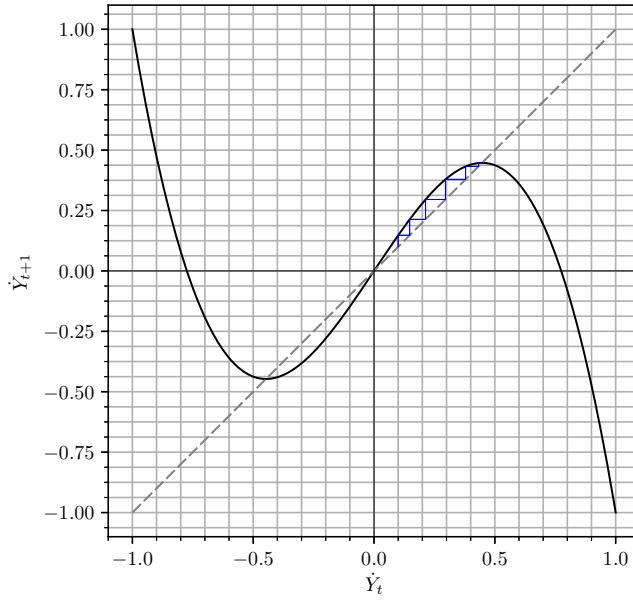


Figure 2.1: Cobweb plot of the multiplier-accelerator model displaying a fixed point at  $\dot{Y} \approx 0.447$ .  $\mu = 1.5$  and  $\dot{Y}_0 = 0.5$

currency, it can be arbitrarily rescaled by selecting different measures of income. We can thus define a new variable:

$$\mu \equiv v - s$$

to arrive at a first-order, single variable function for growth:

$$\dot{Y}_t = \mu \dot{Y}_{t-1} - (\mu + 1) \dot{Y}_{t-1}^3 \quad (2.5)$$

## 2.3 Growth Dynamics

If  $\mu \in [0, 3]$ , this model is bounded within -1 and 1. The mapping has two fixed points:

$$\dot{Y}_t = \dot{Y}_{t-1} = 0, \pm \sqrt{\frac{\mu - 1}{\mu + 1}} \quad (2.6)$$

The first fixed point is stable when  $\mu < 1$  and the second fixed point is stable when  $1 < \mu < 2$ .

When  $\mu > 2$  the fixed point loses stability; however, the double iterate of the function gains a stable fixed point, i.e. a stable 2-cycle forms.

The parameter range of cycle stability decreases as the periodicity of the cycle increases

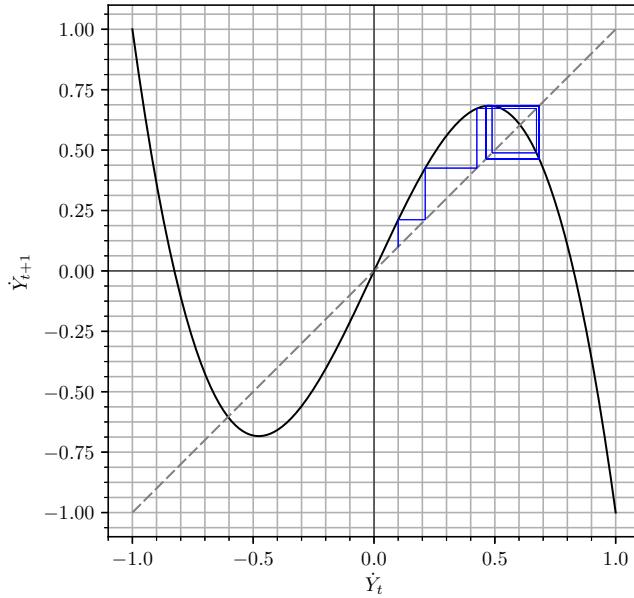


Figure 2.2: Cobweb plot of the multiplier-accelerator model displaying a 2-cycle.  $\mu = 2.15$  and  $\dot{Y}_0 = 0.1$

until  $\mu \approx 2.302$  which is when this mapping reaches the Feigenbaum constant. Once  $\mu$  exceeds this point, the model becomes chaotic; however, it is bound in the quadrant of the original point. The implication of this is that an economy facing growth will continue to encounter growth but an economy that is decaying will always decay. It is possible for the mapping to exit the bounds of its quadrant. The value of  $\mu$  such that the 0 of the mapping is equal to the maximal point of the mapping marks the transition point between bounded chaotic behavior and "unbounded" chaotic behavior (the mapping is still bounded within  $[-1, 1]$ ). This occurs when

$$\mu = \frac{3\sqrt{3}}{2} \approx 2.5981$$

Figure 2.5 displays the transitions this mapping makes between a stable fixed point, a 2-cycle, 4-cycle and so on until chaotic behavior arises. It is important to note though that even in the chaotic region, there exist windows of order. Figure 2.6 displays a stable cycle that arises in one such window of order. This cycle is not bounded within its quadrant, it can thus feature endogenous growth and decay in the same cyclic trajectory, which is a significant distinction from all other cycles displayed thus far.

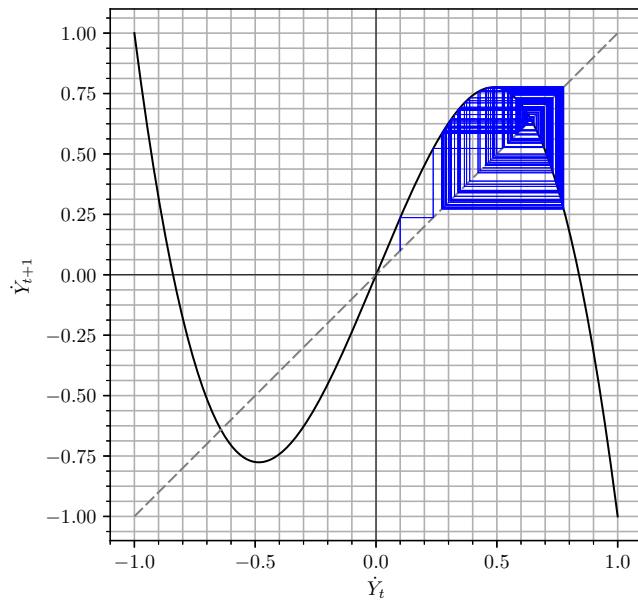


Figure 2.3: Cobweb plot of the multiplier-accelerator model displaying bounded chaos.  
 $\mu = 2.4$  and  $\dot{Y}_0 = 0.1$

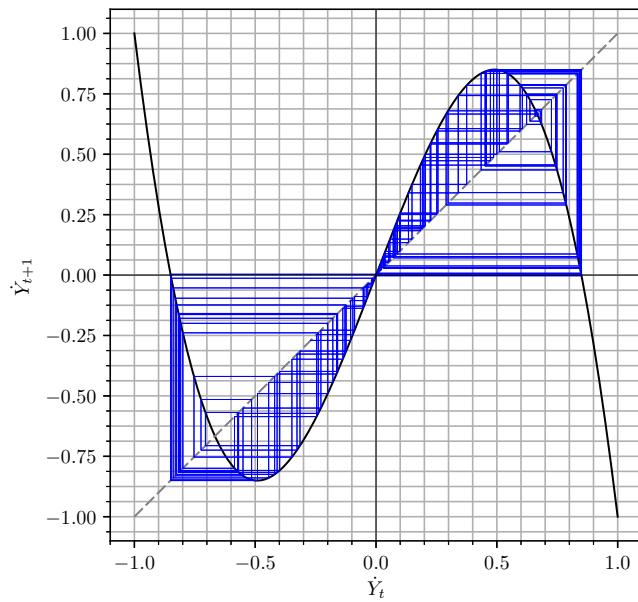


Figure 2.4: Cobweb plot of the multiplier-accelerator model displaying bounded chaos.  
 $\mu = 2.6$  and  $\dot{Y}_0 = 0.1$

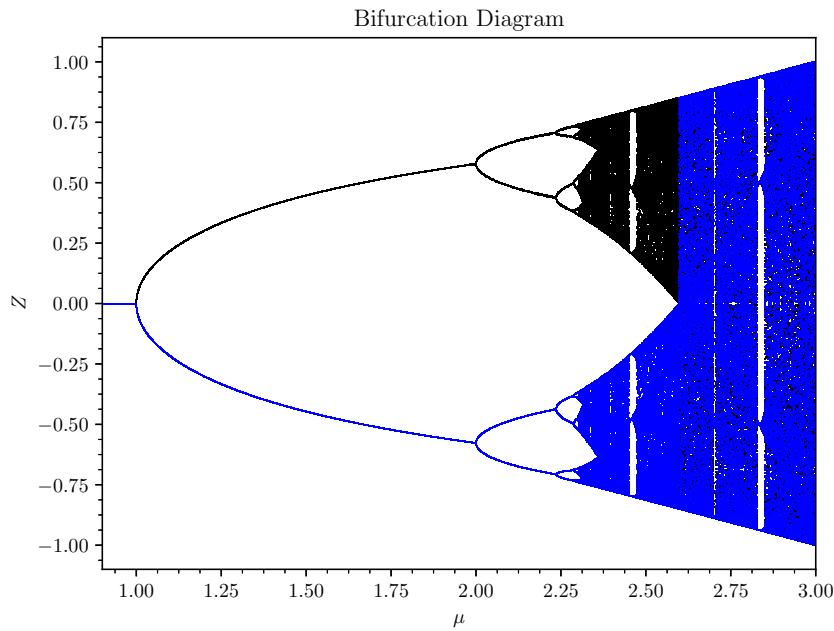


Figure 2.5: Bifurcation diagram of the multiplier-accelerator model varying  $\mu$  between 0.9 and 3. The plot constructed using 0.1 as an initial point is displayed in black and the plot constructed using -0.1 as the initial point is displayed in blue. The last 50 points given a 10000 iteration timeseries are displayed.

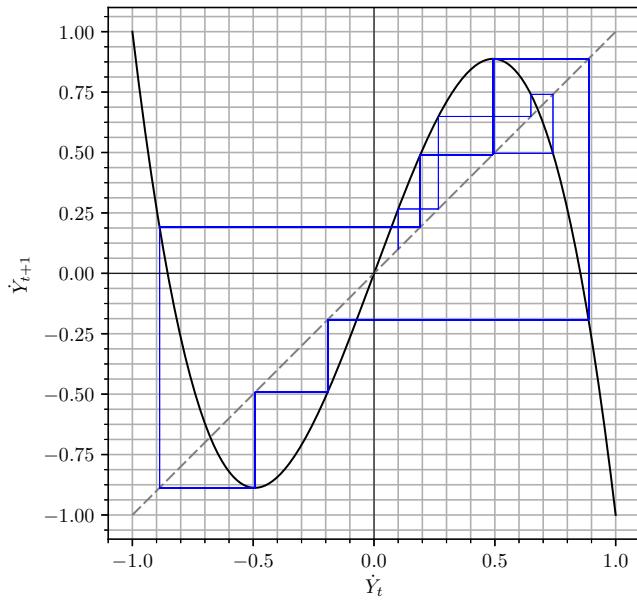


Figure 2.6: Cobweb plot of the multiplier-accelerator model displaying a cycle featuring growth and decay.  $\mu = 2.7$  and  $\dot{Y}_0 = 0.1$

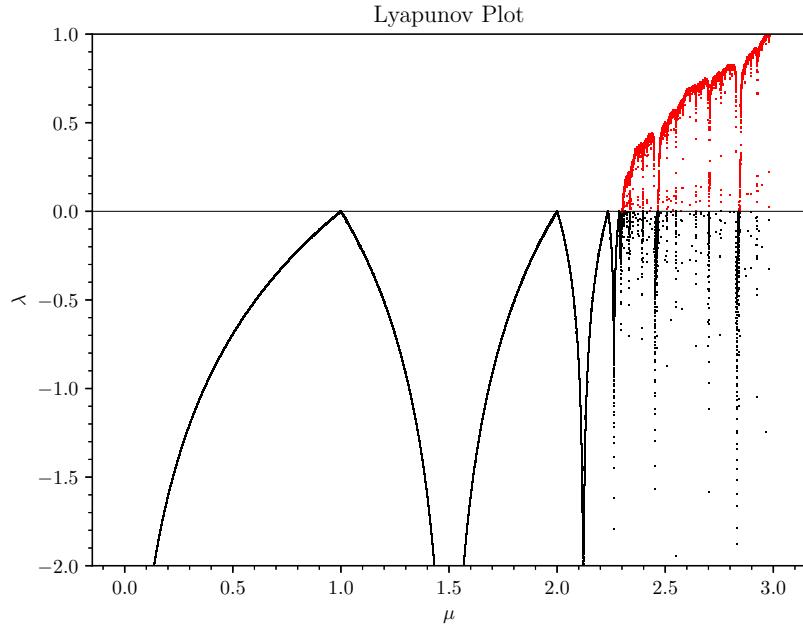


Figure 2.7: Plot of Lyapunov exponent varying  $\mu$  using 0.1 as the initial point. 100,000<sup>th</sup> iteration of the mapping is used to computationally solve for the Lyapunov exponent.

Although the Feigenbaum point provides sufficient proof of the presence of chaos, it is still useful to plot the Lyapunov exponent in order to determine the exact regions of chaotic behavior. Figure 2.7 allows us to explicitly identify the parameter ranges of windows of order in the region of chaos.

### 3. Heterogenous Inventory Cycles by Wegener et al.

#### 3.1 Background

Although nobel prize winning Paul Samuelson was widely credited with providing a mathematical basis to Keynesian economics and popularizing the neoclassical synthesis, he was far from the only economist involved in the movement<sup>13,16</sup>

Another notable Keynesian was Lloyd Metzler who developed a different type of multiplier-accelerator business cycle. Many models developed in the neo-Keynesian era of economics still remain in use but have either been expanded or are used to study specific relationships. Wegener, Westerhoff, and Zaklan expand upon Metzler's inventory cycle model by introducing heterogenous expectations into firm behavior.<sup>17</sup>

Unlike the well known multiplier-accelerator model developed by Samuelson and Hicks, the dynamics of this model are driven by a Lundberg lag and shifts in the heterogenous mix of behaviors in the firms. The Lundberg lag, named after the eponymous economist Erik Lundberg, is based on a difference existing between the quantity produced and the quantity demanded. This is performed by having firms maintain a level of inventory to absorb unexpected increases in consumption. Firms predict consumption using one of two behaviors, a regressive type and an extrapolative type. The extrapolative behavior occurs when a firm predicts that future income will deviate from the steady state. The regressive behavior occurs when firms predict that future income will return to the steady state.

#### 3.2 Model Set-Up

In this business cycle model, the economy is closed and income is determined completely by the quantity of goods produced by firms. Goods are completely homogenized but can be produced for 3 purposes: stock  $S$ , consumption  $U$ , and investment  $I$ . This lets us define income as:

$$Y_t = I_t + S_t + U_t \quad (3.1)$$

Investment is held to be exogenously determined and constant, thus:

$$I_t = \bar{I} \quad (3.2)$$

Consumption adapts to income according to the marginal propensity to consume,  $b$ . As this model follows closely to the framework developed by Metzler,<sup>18</sup> consumption adapts directly to the current income level as follows:

$$C_t = bY_t \quad (3.3)$$

Producers have a desired level of inventory based on the expected level of consumption goods. Thus setting,  $k > 0$ :

$$\hat{Q}_t = kU_t \quad (3.4)$$

where  $Q$  is the level of inventory. In order to achieve this desired level of inventory, firms produce  $S$  amount of output such that:

$$S_t = \hat{Q}_t - Q_{t-1} \quad (3.5)$$

However, the expected level of consumption does not necessarily match the produced level of goods. The realized level of inventory change can thus be determined as:

$$Q_t = \hat{Q}_t - (C_t - U_t) \quad (3.6)$$

The quantity of goods produced for consumption is determined by firms expectations for desired consumption. The extrapolative expectation rule is used by firms that believe consumption levels will continue shifting away from the long-run average quantity. This behavior is interpreted mathematically as:

$$U_t^E = C_{t-1} + c(C_{t-1} - \bar{C}) \quad (3.7)$$

where  $c \geq 0$  denotes the speed at which firms expect consumption to deviate from the long-run level.

The other possible strategy firms can take is to expect consumption to return to the equilibrium level. This is described as the regressive expectation rule and is interpreted mathematically as:

$$U_t^R = C_{t-1} + f(\bar{C} - C_{t-1}) \quad (3.8)$$

where  $0 \leq f \leq 1$  denotes the expected adjustment speed to the long-run level.

As this economy only involves two expectation rules, the aggregate expected level of consumption is a simple weighted average:

$$U_t = w_t U_t^E + (1 - w_t) U_t^R \quad (3.9)$$

The weight is defined as a function of consumption level which allows firms to switch their predictive behavior based on the current consumption level. Intuitively, as consumption deviates further from equilibrium, more firms will believe that the boom or slump will end and adjust in accordance. Likewise, when consumption is close to equilibrium, firms will believe it to be more accurate to extrapolate consumption. This behavior is modelled by the function:

$$w_t = \frac{1}{1 + d(\bar{C} - C_{t-1})^2} \quad (3.10)$$

where  $d$  is proportional to the popularity of the regressive rule.

Taking these rules in aggregate, income is a second-order nonlinear iterated mapping. Condensing the model, we arrive at the mapping:

$$Y_t = U_t + kU_t - (1 + k)U_{t-1} + C_{t-1} + \bar{I} \quad (3.11)$$

This gives a single fixed point of:

$$\bar{Y} = \frac{1}{1 - b} \bar{I} \quad (3.12)$$

It is apparent that, as  $\bar{Y}$  is constant, the theoretical set-up of the model is only valid if there is no long-term growth, that is to say income must be bounded. This is because the predictive mechanism used by firms is reliant on the difference between the lag in consumption and the steady-state level of consumption. If consumption were to consistently grow, then this difference would also grow over time, eliminating any effective purpose in providing heterogenous expectations as firms would practically unanimously switch to the extrapolative mechanism.

### 3.3 Analysis of Income Dynamics

In the example trajectory given above, income clearly followed bounded, cyclic behavior. As seen with the analysis of the logistic map however, there are other types of long-term behavior possible. Given the restrictions on parameters, the steady-state level of income is stable if:

$$k < \frac{1 - b - bc}{b(1 + c)} \quad (3.13)$$

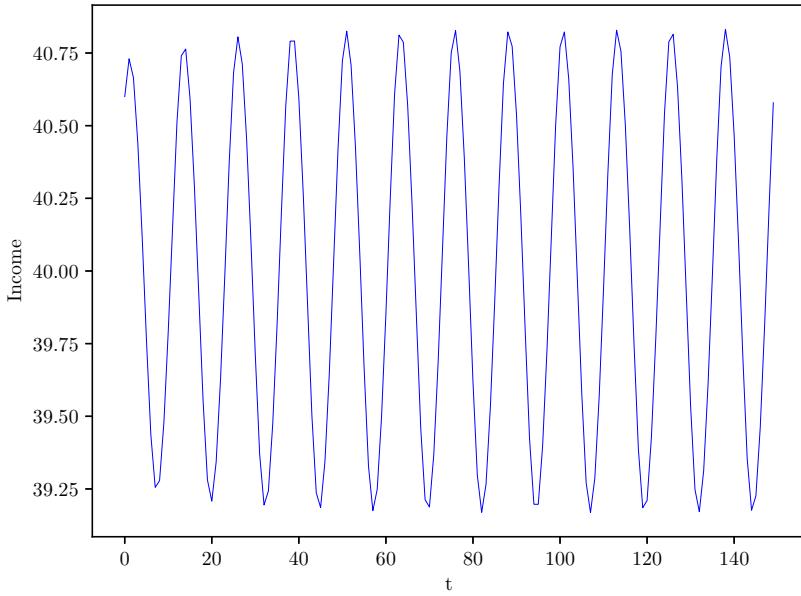


Figure 3.1: Timeseries plot of inventory cycle as described by Wegener et al.<sup>17</sup> Simulation is determined given  $Y_0 = 40.6$ ,  $U_0 = 30.3$ , and  $\bar{I} = 10$ . The parameters of the model are:  $b = 0.75$ ,  $c = 0.3$ ,  $d = 1.0$ ,  $f = 0.1$ ,  $k = 0.1$ .

The proof for this can be seen in the appendix of Wegener et al.<sup>17</sup> The stability of the steady state is only dependent on  $b$ ,  $k$ , and  $c$ , thus the behavior of the regressive expectation rule and the ratio of firms behaving with each expectation rule do not affect the stability of this fixed point. This can be seen graphically using bifurcation diagrams of the parameters.

There is a Neimark-Sacker bifurcation where  $b \approx 0.65$  which marks a transition point from a stable fixed point to a bounded cycle. However, as  $b$  approaches 1, the bifurcation diagram does not graphically feature cycles of well defined order. Varying  $c$  and  $k$  alters the behavior of the cycles as seen in Figure 3.3 and 3.4. Altering  $d$  or  $f$  affects the magnitude of the inventory cycles but does not have any effect on the presence of the cycles.

The bifurcation diagrams varying the  $c$ ,  $k$ ,  $d$ , and  $f$  parameters appear to display clear cyclic behavior. However, this is actually a product of the graphical display of the bifurcation diagram. The diagram is constructed by iterating the mapping over 1000 time periods and displaying the results of the last 50 time-periods. This results in the appearance of 50-cycles over the span of the bifurcation diagram; however, reducing or increasing the amount of iterations to display likewise changes the apparent periodicity of

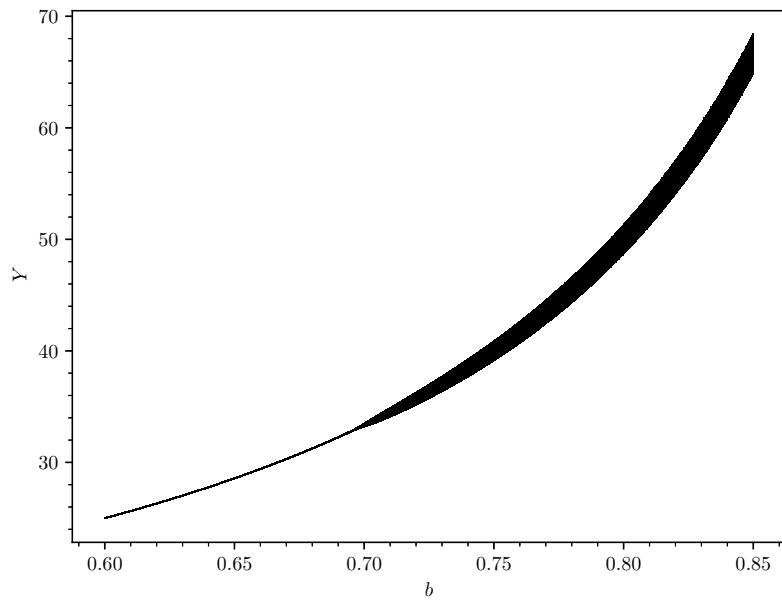


Figure 3.2: Bifurcations diagram varying the parameter  $b$  over the range  $[0.2, 0.85]$  holding all other parameters and initial conditions as described in Figure 3.1. The simulation was allowed to run for 1000 iterations and the last 50 points are captured in the diagram.

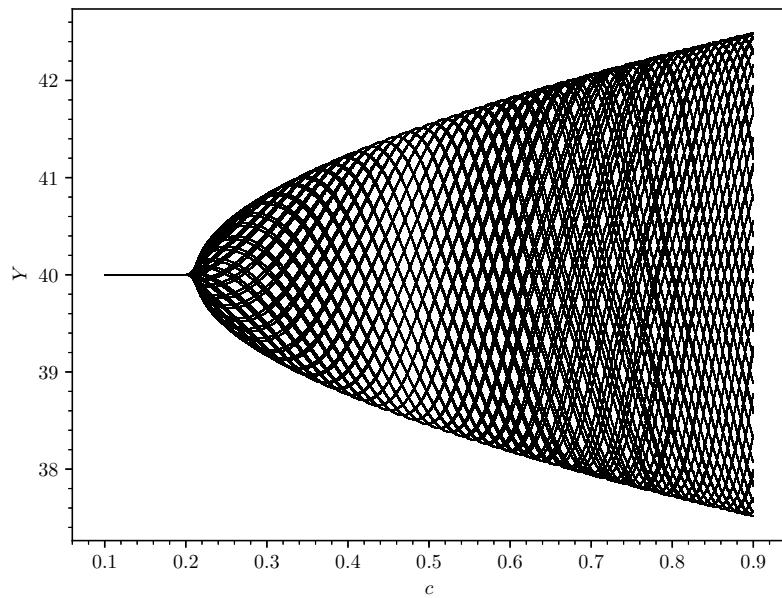


Figure 3.3: Bifurcations diagram varying the parameter  $c$  over the range  $[0.1, 0.9]$  holding all other parameters and initial conditions as described in Figure 3.1. The simulation was allowed to run for 1000 iterations and the last 50 points are captured in the diagram.

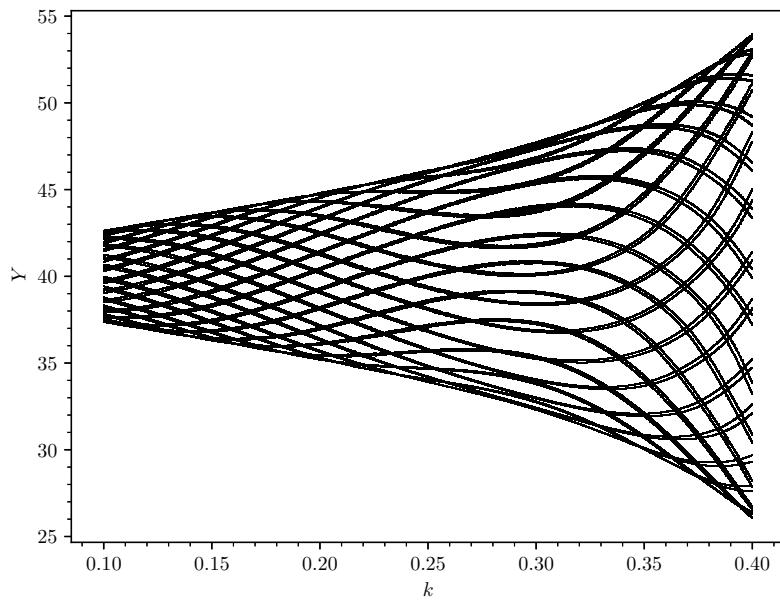


Figure 3.4: Bifurcations diagram varying the parameter  $k$  over the range  $[0.1, 0.4]$  holding all other parameters and initial conditions as described in Figure 3.1. The simulation was allowed to run for 1000 iterations and the last 50 points are captured in the diagram.

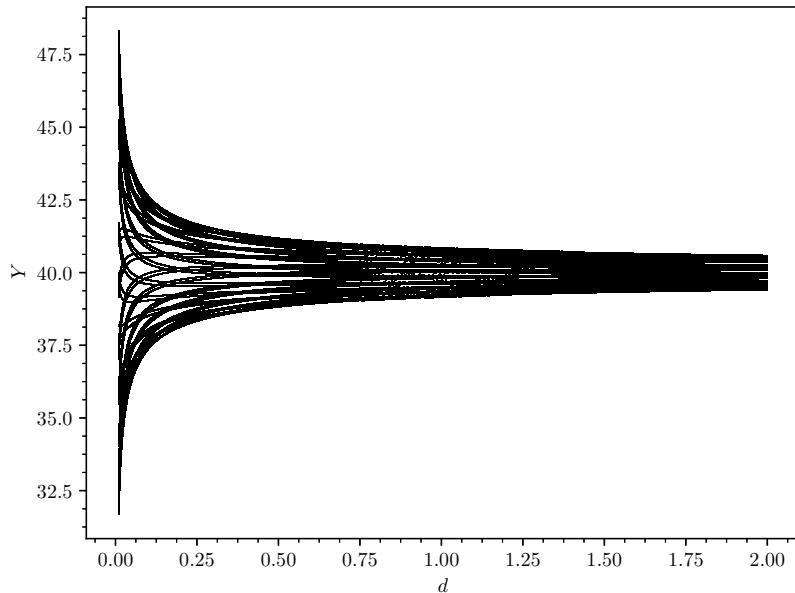


Figure 3.5: Bifurcations diagram varying the parameter  $d$  over the range  $[0.01, 2.0]$  holding all other parameters and initial conditions as described in Figure 3.1. The simulation was allowed to run for 1000 iterations and the last 50 points are captured in the diagram.

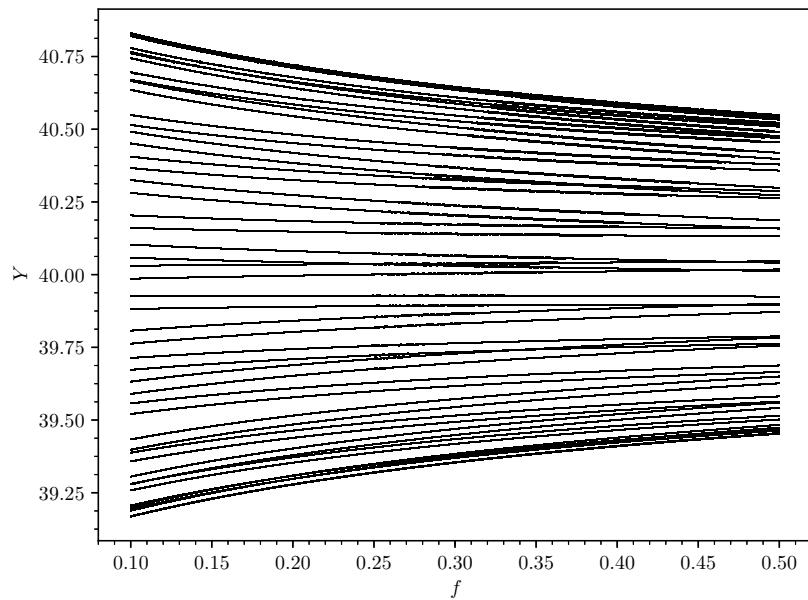


Figure 3.6: Bifurcations diagram varying the parameter  $f$  over the range  $[0.1, 0.5]$  holding all other parameters and initial conditions as described in Figure 3.1. The simulation was allowed to run for 1000 iterations and the last 50 points are captured in the diagram.

the cycle. This implies the presence of more points in the long-run trajectories than is visible in the diagram.

## 4. Inventory Cycles with Averaged Expectations, Endogenous Investment, and Consumption Lag

### 4.1 Model Set-Up

Metzler's and Wegener's model show that it is possible for endogenous business cycles to be induced through inventory cycles and the failure of firms to accurately predict future consumption. These models were designed to focus on the effects of firm expectations with Wegener's contribution consisting of introducing a heterogenous expectation rule that allows firms to switch behavior based on the state of the economy. Both of these models however simplify other aspects of the economy that feature more complex behavior in other business cycle models.

The inventory cycle model features 3 main factors of production: predicted consumption, investment, and inventory. Metzler and Wegener holds investment as an exogenously determined constant; however, this is both unrealistic and does not allow for long term endogenous growth. To incorporate a mechanism for endogenously adjusting investment, we take inspiration from the multiplier-accelerator model presented in Chapter 2.

The problem with the cubic investment function used in that model is that it leads to unbounded behavior in the extremes, much like the linear function. Puu resolves this by ensuring that income growth is bounded by  $[-1, 1]$  but this is not the case for our model. We instead want a function that features similar curvature and behavior as the cubic function over moderate values but flattens when income change is of significant magnitude. This can be accomplished with the following function:

$$I_t = \frac{\frac{Y_{t-1} - Y_{t-2}}{v}}{(\frac{Y_{t-1} - Y_{t-2}}{v})^4 + q} \quad (4.1)$$

This function gives the backbending behavior of the cubic function without the exploding results.

The behavior of this function qualitatively resembles that of the linear-cubic function described in by Puu when  $v, q > 0$ ; however,  $\lim_{(Y_{t-1} - Y_{t-2}) \rightarrow \infty} I_t = 0$ . This function is at

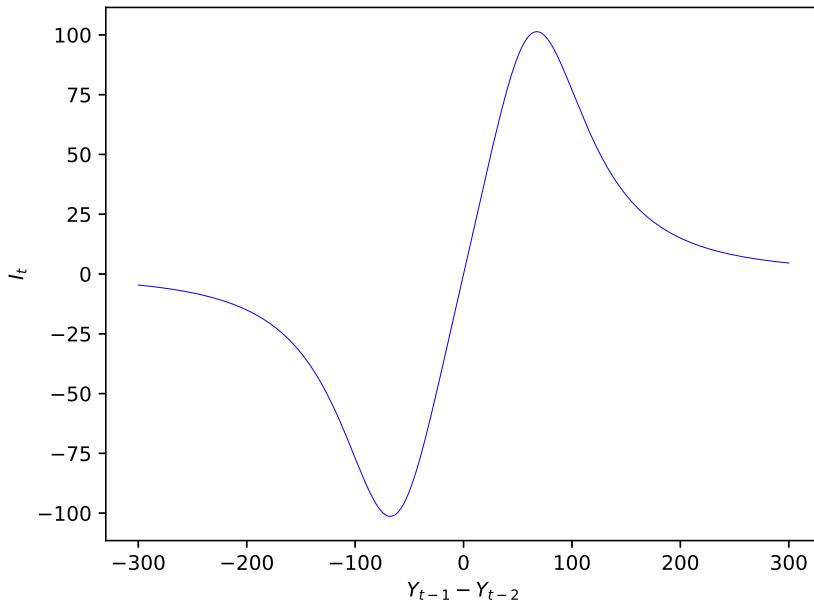


Figure 4.1: Plot of investment curve when  $v = 500$  and  $q = 0.001$ .

its absolute maximum and minimum when:

$$Y_{t-1} - Y_{t-2} = \pm \frac{q^{1/4}v}{3^{1/4}}$$

Thus giving a maximal or minimal investment of:

$$I_t = \frac{3^{3/4}}{4q^{3/4}}$$

Intuitively, larger values of  $v$  make the function "wider", increasing the range over  $Y_{t-1} - Y_{t-2}$  where  $I_t$  is not effectively 0. Smaller values of  $q$  increases the magnitude of the maximum and minimum of the function, effectively making it taller.  $v$  and  $q$  cannot be directly attributed to any direct economic phenomena; however, their effect on the function can be interpreted analogously to the properties of the linear-cubic function. The "height" of the function determines the potential magnitude of investment and the "width" determines the rate at which investment, both private and public, react to changes in income. Fig 4.3 shows that the FWHM of the investment curve has a nearly positive, linear relationship with both  $v$  and  $q$  over tested values of  $v$  and  $q$ .

Metzler describes the existence of two important lags in the study of Keynesian models. The Robertson lag is characterized by making current consumption a function of

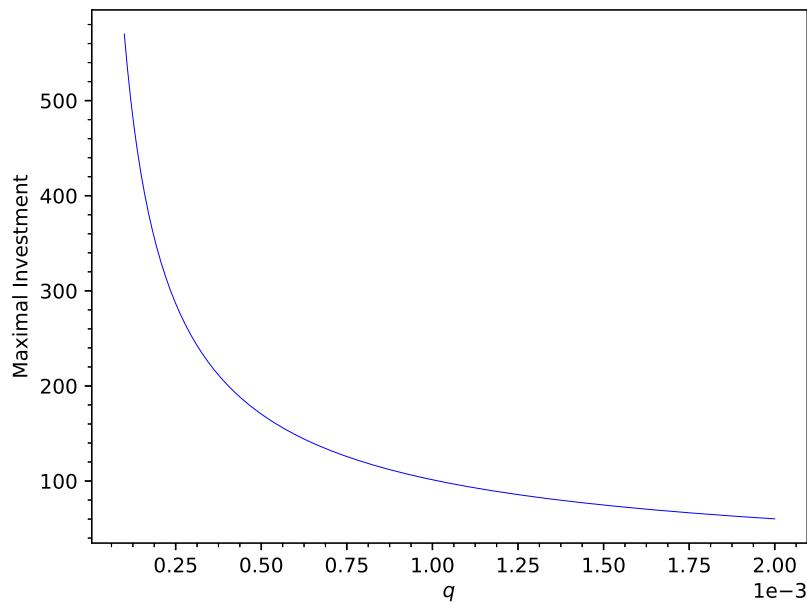


Figure 4.2: Plot of maximal investment as  $q$  varies between 0 and 2000.

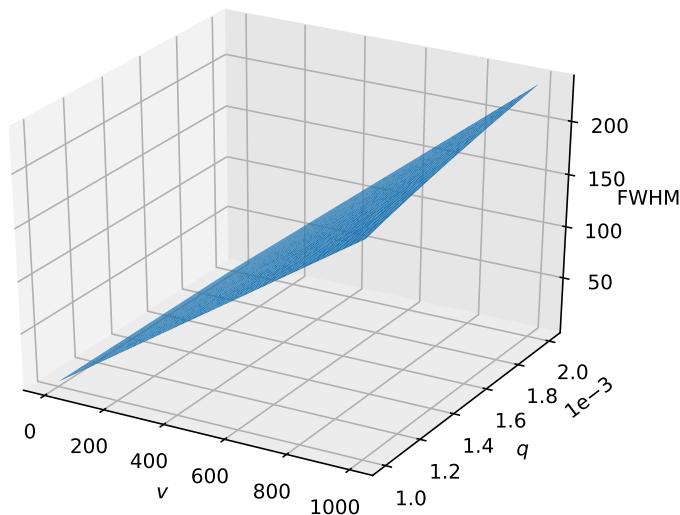


Figure 4.3: Plot of the full width at half maximum (FWHM) of the investment curve as  $v$  varies between 1 and 2000 and  $q$  varies between 0.001 and 0.002

past income, i.e. consumption behavior lags behind current income. The Lundberg lag however concerns a discrepancy between the income level and the production decision of firms.<sup>18</sup> These lags are named after the two economists D. H. Robertson and Erik Lundberg who developed models that contained only their eponymous lag type. Although both lags are likely to exist in reality, most models only incorporate one lag due to the increased complexity associated with it. Metzler and Wegener make use of a Lundberg lag by making income a function of the predicted level of consumption as opposed to actual consumption. Tönu Puu's model makes use of a Robertson lag in order to induce endogenous business cycles. Metzler himself does not claim that the Lundberg sequence is any more realistic than the Robertson sequence. Whichever lag has a longer time-period can be treated as of being greater importance but Metzler actually proposes a variety of scenarios that present contradicting conclusions. Suppose that firms decided their behavior on a quarterly basis but consumers altered their spending behavior with every paycheck, then it would be appropriate to treat the Robertson lag as being of 0 length, i.e. nonexistent. If consumers revise their spending behavior every 6 months to a year, then it would actually be more realistic to include a non-zero Robertson lag while minimizing the Lundberg lag.

For the purposes of this model, we will include a non-zero Lundberg and Robertson lag. The consumption function is treated exactly as presented in Puu:

$$C_t = (1 - s)Y_{t-1} + sY_{t-2} \quad (4.2)$$

As the economy is also making use of a Robertson lag, income is not directly a function of consumption as it is 2. Rather, income is viewed from a production standpoint. This is achieved by explicitly defining a predicted level of consumption. Wegener and Metzler use an equation similar to the form:

$$U_t = C_{t-1} + \eta(C_{t-1} - C_{t-2})$$

However, this assumes that firms have no way to adapt their coefficient of expectation  $\eta$ .

A new predictive mechanism requires that firms have bounded rationality. If firms possessed perfect rationality, then they would be able to perfectly predict consumption levels as consumption is based on lagged income, thus if we wish to maintain our Robertson lag, firms must still have bounded rationality. Another mechanism that firms can use that maintains better "memory" is that of an average.

$$U_t = \frac{C_{t-1} + C_{t-2} + C_{t-3}}{3} \quad (4.3)$$

This formulation is an average of the consumption levels of the past 3 time periods. The choice of three time periods is relatively arbitrary and higher order difference equations of the same form can be used; however these higher order formulations can result in a less mathematically tractable model.

Inventory production proceeds as seen in Chapter 2 with firms producing  $S_t$  explicitly to maintain inventory at optimal levels:

$$S_t = kU_t - Q_{t-1} \quad (4.4)$$

where  $Q_t$  is the level of inventory maintained at the end of time  $t$  and  $k \in [0, 1]$ . This can be solved for as the sum of the previous inventory level, production intended for inventory, and the difference between production intended for consumption and the actual consumption level:

$$Q_t = Q_{t-1} + S_t + (U_t - C_t) \quad (4.5)$$

Income level, or output, can thus be written as a sum of production:

$$Y_t = I_t + S_t + U_t \quad (4.6)$$

However, as income has the capability of sustained growth under this model, it is preferable to analyze the rate of change of income in this model. In Appendix A, we show that the growth rate of the economy can be interpreted as a single variable, 6th order difference equation on  $\dot{Y}$ :

$$\begin{aligned} \dot{Y}_t = & \frac{\frac{\dot{Y}_{t-1}}{v}}{\left(\frac{\dot{Y}_{t-1}}{v}\right)^4 + q} - \frac{\frac{\dot{Y}_{t-2}}{v}}{\left(\frac{\dot{Y}_{t-2}}{v}\right)^4 + q} + \\ & \frac{k+1}{3} \left[ (1-s)(\dot{Y}_{t-2} - Y_{t-5}) + s(\dot{Y}_{t-3} - Y_{t-6}) \right] + (1-s)\dot{Y}_{t-2} + s\dot{Y}_{t-3} \end{aligned} \quad (4.7)$$

## 4.2 Growth Dynamics

The long run dynamics of the model are highly dependent on the choice of initial conditions. Simulation of the model require a choice of 6 initial values of income change in addition to the parameter choice. If the 6 initial values are equivalent, all future iterations of the mapping are equal to this choice and the model remains in a stable steady state. It is both more interesting and realistic then, to consider cases where there is variation in initial income growth. Figure 4.4 displays a possible trajectory of income growth. Under

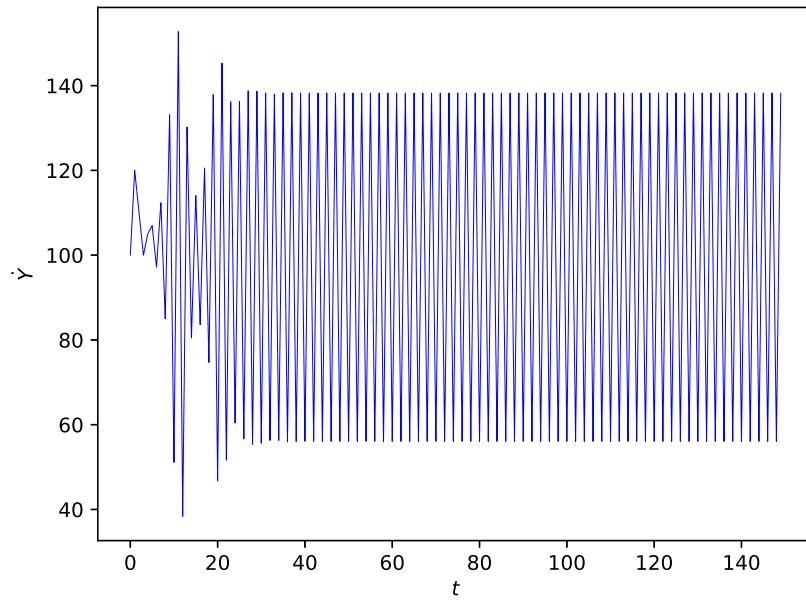


Figure 4.4: Timeseries plot of income growth rate over 150 iterations.  $s = 0.6$ ,  $k = 0.3$ ,  $v = 500$ ,  $q = 0.001$ . Initial values of  $\dot{Y}$  are: 100, 120, 110, 100, 105, 107

the parameters listed, the trajectory clearly behaves in a stable, cyclic manner within 40 iterations.

There do not exist techniques to solve generalized 6th order difference equations; however it is possible to computationally solve for the long run dynamics of the system. The bifurcation diagram displayed in Figure 4.5 displays the long-run behavior of the model as  $s$  is varied. The methodology of the computational analysis of the bifurcation points and the results of this analysis are displayed in detail in Section E.5 and Section E.6, respectively. Long-run values are rounded to their nearest whole number for the purposes of this computational analysis although it is possible to account for higher levels of numerical precision using the analyzer. The mapping displays a stable 2-cycle when  $s \in (0.65, 0.53)$ . This implies the presence of a stable business cycle when the marginal propensity to save is moderately high; however, the behavior appears chaotic when the marginal propensity to save approaches either extreme.

The bifurcation diagram varying  $k$  seen in Figure 4.6 qualitatively appears much simpler than Figure 4.5. This diagram displays a bifurcation point when  $k \approx 0.84837$ . The model displays stable 2-cycle behavior until this point although the system continues to display windows of ordered cycles. It thus appears that although increasing the desired proportional level of inventory does increase the long-run level of cyclic growth, it does

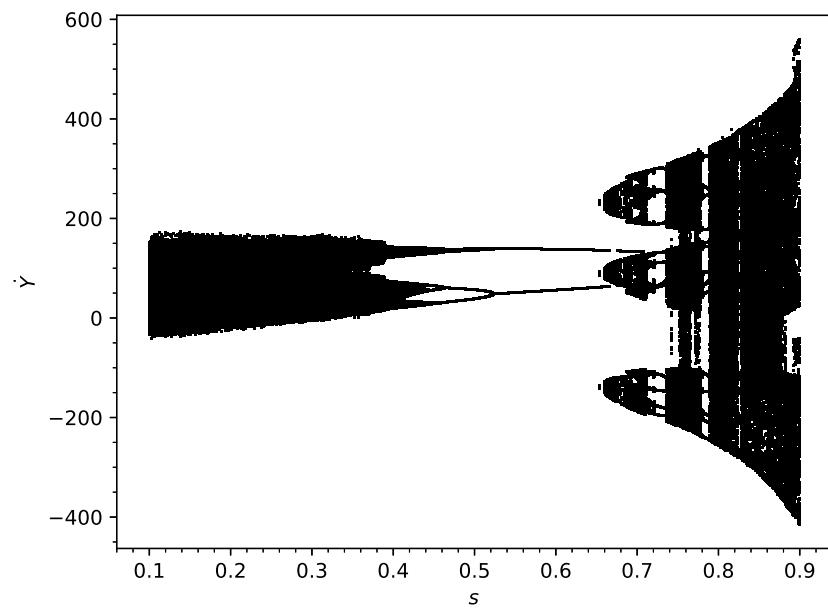


Figure 4.5: Bifurcation diagram varying  $s$  between 0.1 and 0.9. Initial conditions and other parameters are held constant as described in Figure 4.4.

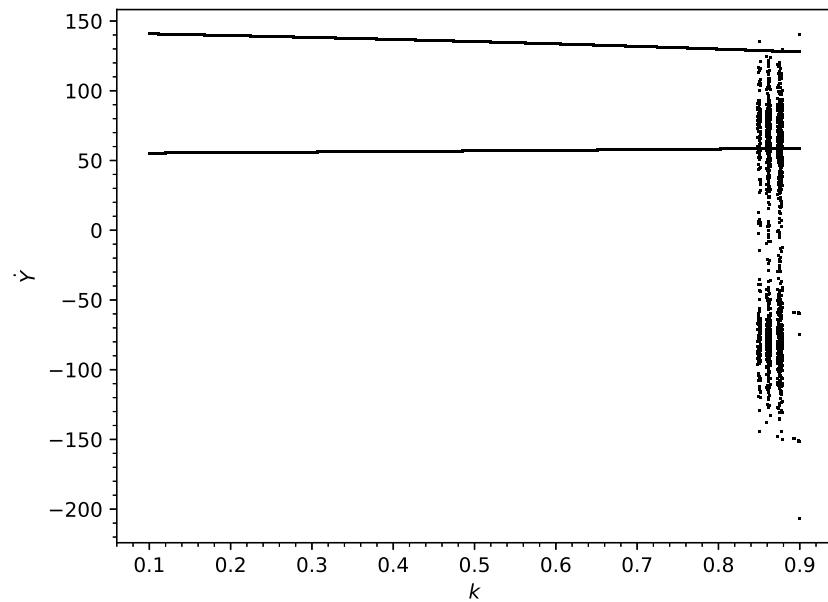


Figure 4.6: Bifurcation diagram varying  $k$  between 0.1 and 0.9. Initial conditions and other parameters are held constant as described in Figure 4.4.

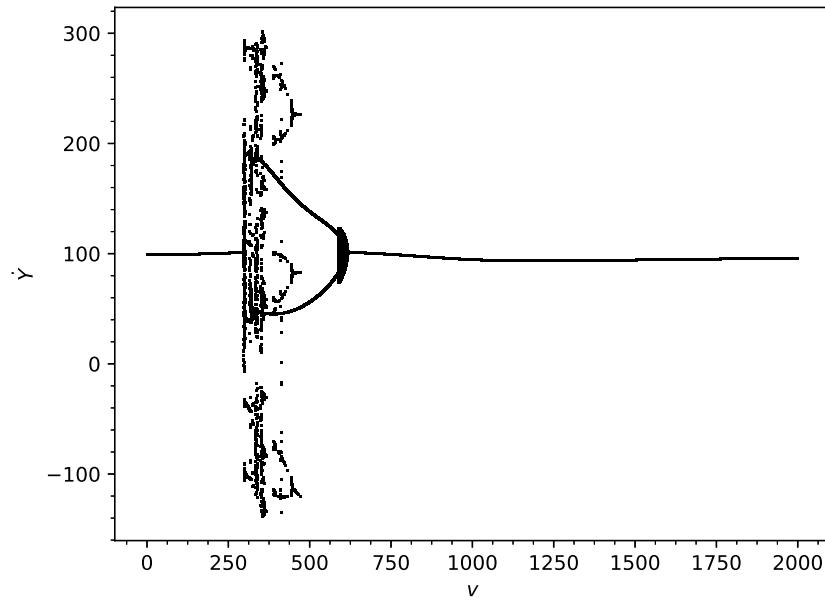


Figure 4.7: Bifurcation diagram varying  $v$  between 1 and 2000. Initial conditions and other parameters are held constant as described in Figure 4.4.

not affect the topology until  $k$  is unrealistically large.

Figure 4.7 exhibits stable, fixed growth when  $v \in (1, 297.338) \cup (617.104, 2000)$  which is the upper bound of what was computationally determined. The model also displays a stable 2-cycle when  $v \in (472.173, 589.757)$ . Likewise, Figure 4.8 has a bifurcation point at  $q \approx 0.0015997$  and  $q \approx 0.0015976$  which mark transitions from a stable, fixed point to a 2-period cycle to a 3-period cycle (more bifurcation points exist within the diagram and are more easily viewed by narrowing the parameter variation range, this can be seen in Figure 4.22). As stated previously,  $v$  and  $q$  individually lack an economic interpretation, but together they approximate the dynamics of a linear-cubic investment curve while incorporating bounded, asymptotic behavior.

These bifurcation diagrams do not tell the complete story of this model however for any given set of parameters. As stated previously, the choice of constant initial conditions will result in a constant trajectory of income growth. However, it is also trivial to show the existence of other trajectories by creating a timeseries holding the parameters constant but shifting the initial conditions.

The timeseries displayed in Figure 4.4 and Figure 4.9 share the same parametrization but they clearly display distinct dynamics and are centered around different growth rates. It can be helpful to think of each initial value as a parameter for the purposes of analysis,

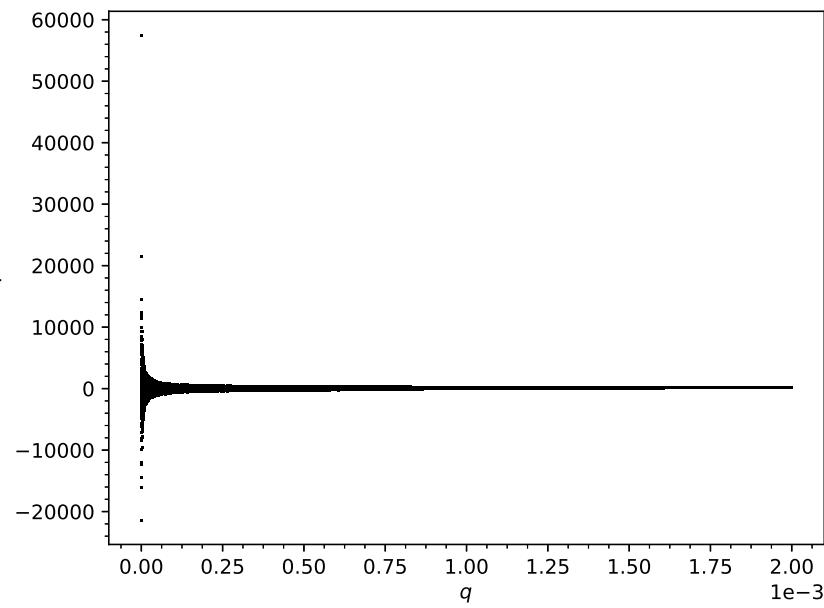


Figure 4.8: Bifurcation diagram varying  $q$  between 0 and 0.0001. Initial conditions and other parameters are held constant as described in Figure 4.4.

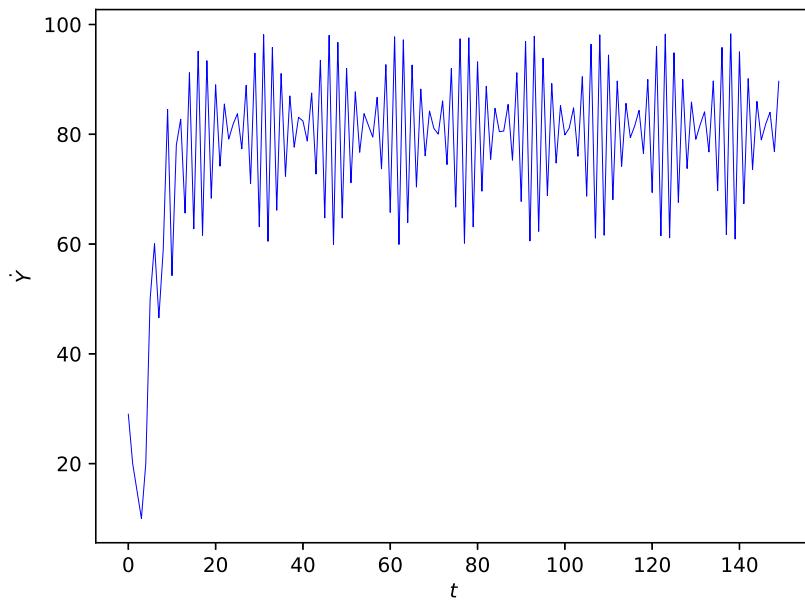


Figure 4.9: Timeseries plot of income growth rate over 150 iterations. Parameter choice is identical to that of Figure 4.4. Initial values of  $\dot{Y}$  are: 29, 20 , 15 , 10, 20 ,50,

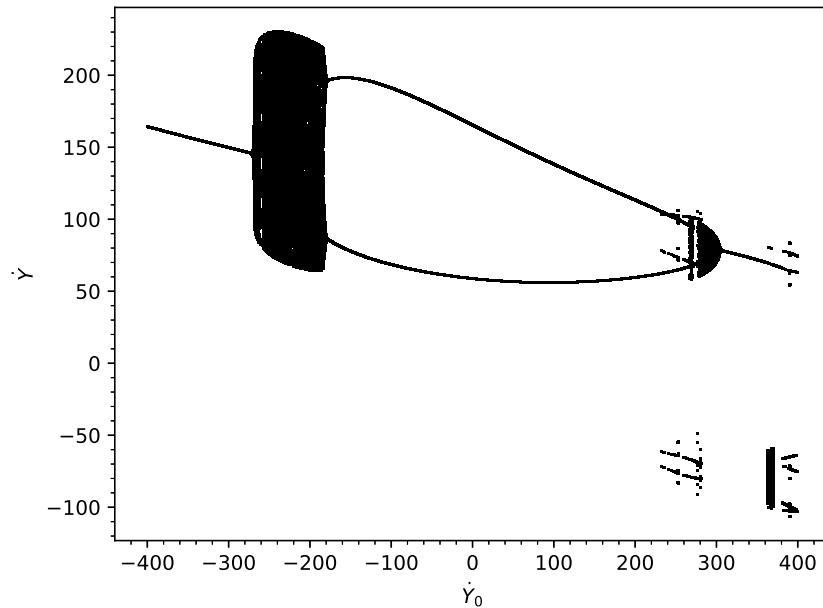


Figure 4.10: Bifurcation diagram varying  $\dot{Y}_0$  between -400 and 400. All other parameters and initial conditions kept as described in Figure 4.4

thereby allowing us to create bifurcation diagrams in order to identify their effects on the long-run behavior of the model.

This diagram shows several clearly defined regions of stable, ordered behavior. There is a bifurcation when  $Y_0 \approx -267.986$  that causes the dynamics of the mapping to shift away from fixed, stable growth. There is also a region where stable 2-cycles exist when  $Y_0 \in (-180.378, 231.903)$  before the emergence of higher order cycles.

Figure 4.11 is also relatively simple to analyze, featuring a fixed growth rate until the bifurcation point where  $Y_1 \approx -99.010$ . A region of stable 2-cycles also exist when  $Y_1 \in (-48.205, 220.542)$ . Fixed growth recommences when  $Y_1 \approx 243.104$  before shifting back into a 2-cycle region when  $Y_1 \approx 313.351$ .

Figure 4.12 is qualitatively almost identical to the diagram displayed in Figure 4.11. The model displays fixed growth until the bifurcation point when  $Y_2 \approx -109.011$ . A region of stable 2-cycles exists when  $Y_2 \in (-58.206, 210.541)$ . Fixed growth recommences after the bifurcation point when  $Y_2 \approx 233.103$  before shifting back into stable 2-cycles when  $Y_2 \approx 303.350$ .

Figure 4.13 possess many regions of clearly visible, ordered cyclic behavior, of note however is presence of stable 2-cycles when  $Y_3 \in (19.162, 270.867)$ .

Figure 4.14 displays fixed behavior when  $\dot{Y}_4$  is very large or small compared to the

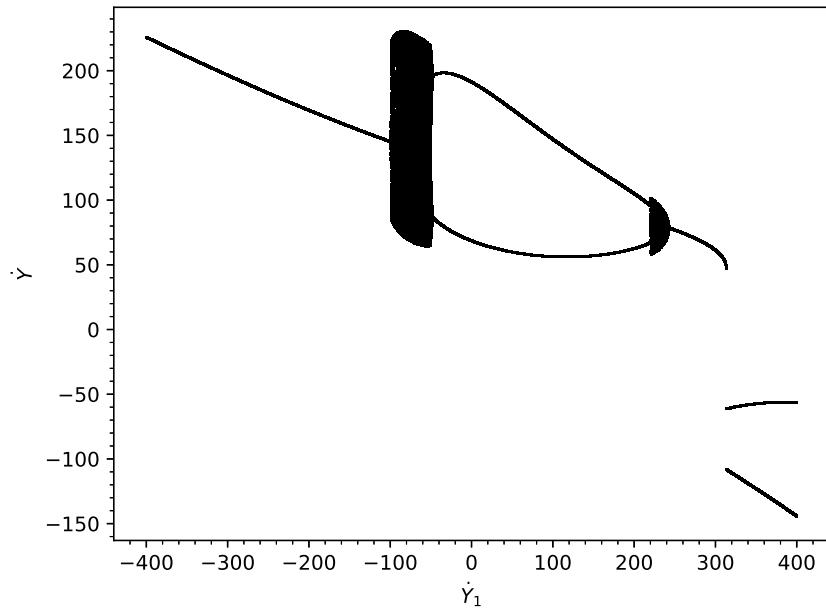


Figure 4.11: Bifurcation diagram varying  $\dot{Y}_1$  between -400 and 400. All other parameters and initial conditions kept as described in Figure 4.4

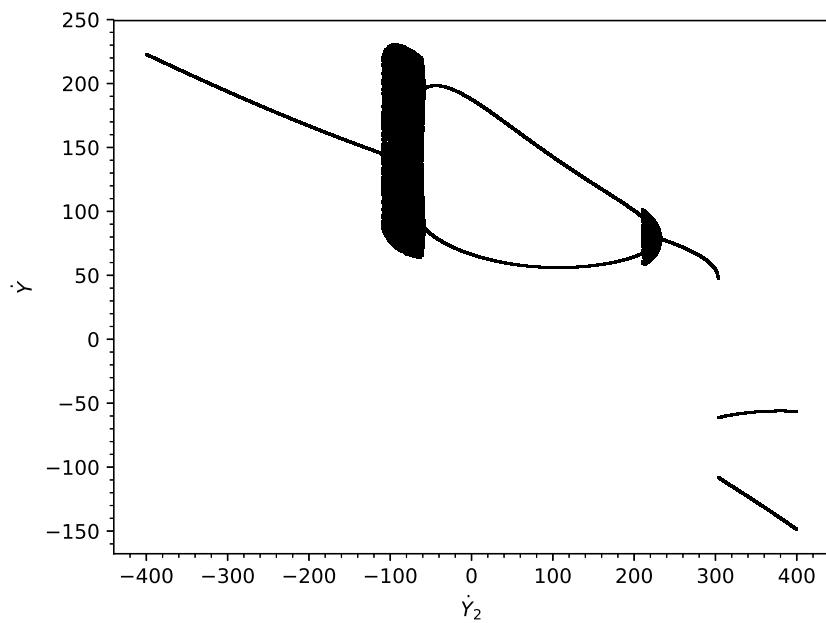


Figure 4.12: Bifurcation diagram varying  $\dot{Y}_2$  between -400 and 400. All other parameters and initial conditions kept as described in Figure 4.4

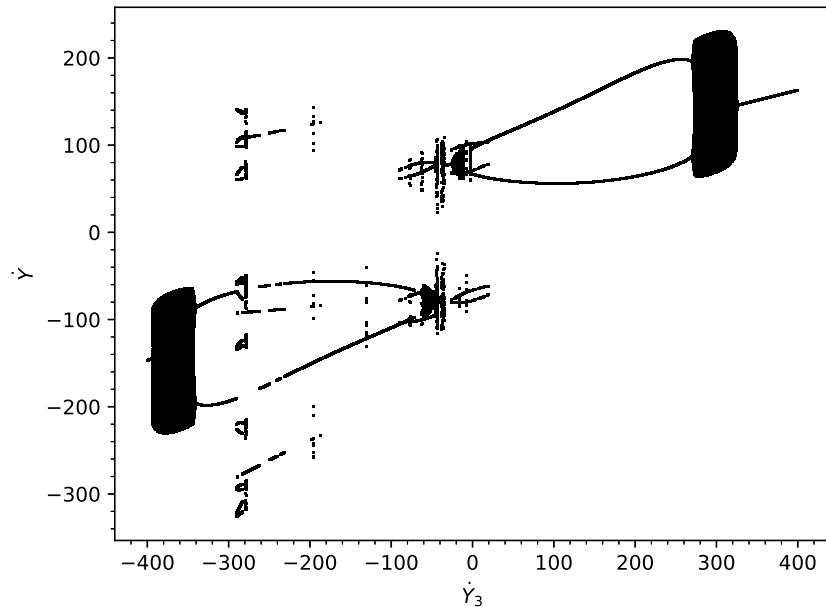


Figure 4.13: Bifurcation diagram varying  $\dot{Y}_3$  between -400 and 400. All other parameters and initial conditions kept as described in Figure 4.4

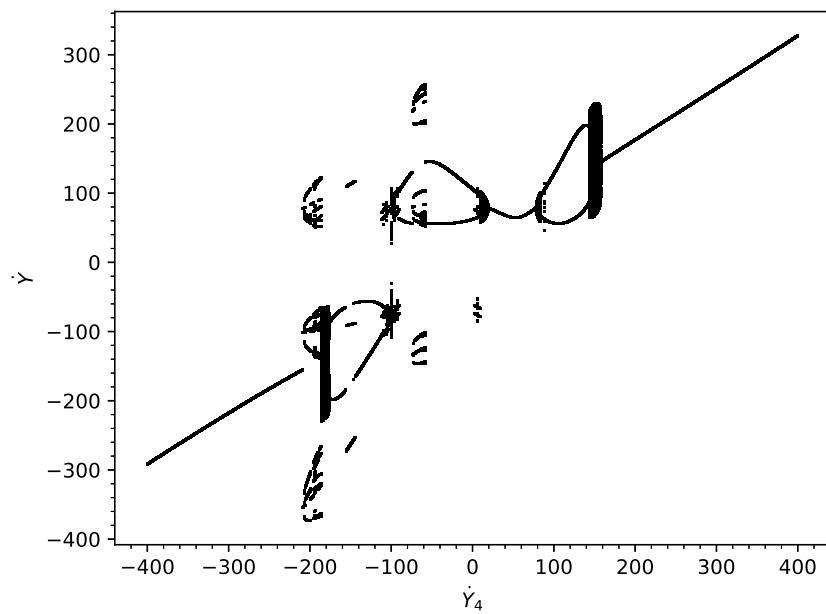


Figure 4.14: Bifurcation diagram varying  $\dot{Y}_4$  between -400 and 400. All other parameters and initial conditions kept as described in Figure 4.4

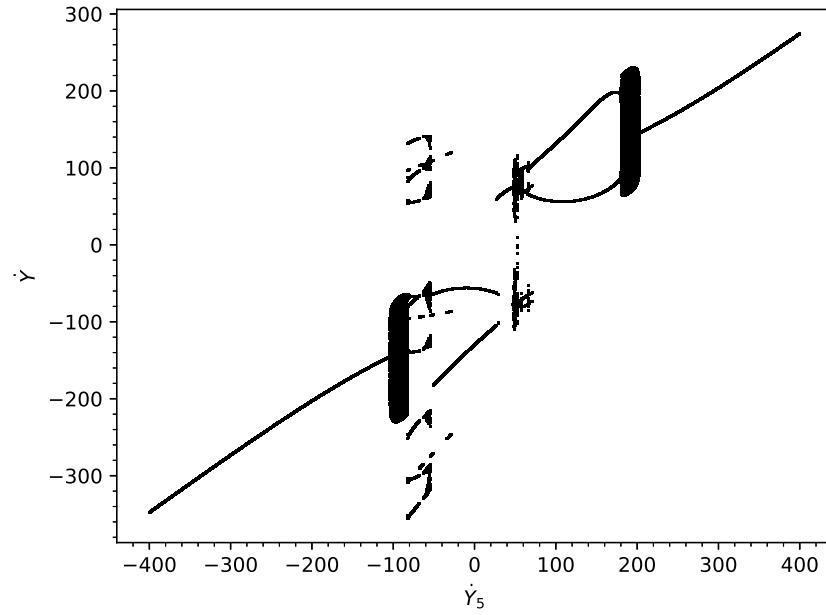


Figure 4.15: Bifurcation diagram varying  $\dot{Y}_5$  between -400 and 400. All other parameters and initial conditions kept as described in Figure 4.4

other initial variables. When their values are more moderate, the long-run behavior of the model can be a 2-cycle such as when  $\dot{Y}_4 \in (-144.054, -111.571)$  or  $\dot{Y}_4 \in (87.880, 143.494)$ . Higher order cycles and chaotic behavior is also present in this moderate range.

Figure 4.15 also displays fixed behavior when  $\dot{Y}_5$  is very large or small compared to the other initial variables. 2-cycles are present when  $\dot{Y}_5 \in (-28.843, 27.243) \cup (71.167, 179.899)$  although chaotic behavior and higher order cycles are also present.

### 4.3 Chaotic Behavior in Growth

Bifurcation analysis of this type allows us to numerically determine the presence and location of bifurcation points separating arbitrary period cycles. However, this is insufficient to determining if varying the parameters of the model result in chaotic behavior and where exactly this transition occurs. The existence of chaos lends credence to the assumption of bounded rationality assigned to firms. If the model always features stable cycles, then it would only be logical that firms would eventually be able to adapt their predictive mechanism to predict this ordered behavior. The effect this shift in expectation rule would have on future dynamics is beyond the scope of this paper but could be performed by removing the Lundberg lag after the system has evolved over time. the

presence of chaos means the model is sufficiently sensitive to initial conditions that firms would not be able to accurately determine the future state of the market as under a chaotic system, knowing the approximate current state of the economy does not inform firms of the approximate future state of the economy.

The method described in the introduction to solve for the Lyapunov exponent is applicable to 1-dimensional systems. Although our system is formally a 1-dimensional, 6th order difference equation, existing methods of analysis are much more accommodating to single-order mappings. We thus begin by rewriting the model as a 1st-order difference equation with 6 variables. This can be done by creating and defining the variables  $\dot{Y}_{t-1} = \dot{A}_t$ ,  $\dot{A}_{t-1} = \dot{B}_t$ ,  $\dot{B}_{t-1} = \dot{C}_t$ ,  $\dot{C}_{t-1} = \dot{D}_t$ , and  $\dot{D}_{t-1} = \dot{G}_t$ . This allows us to define an equivalent model:

$$\begin{aligned}\dot{Y}_t &= f(\dot{Y}_{t-1}, \dot{A}_{t-1}, \dot{B}_{t-1}, \dot{C}_{t-1}, \dot{D}_{t-1}, \dot{G}_{t-1}) \\ \dot{A}_t &= g(\dot{Y}_{t-1}, \dot{A}_{t-1}, \dot{B}_{t-1}, \dot{C}_{t-1}, \dot{D}_{t-1}, \dot{G}_{t-1}) \\ \dot{B}_t &= h(\dot{Y}_{t-1}, \dot{A}_{t-1}, \dot{B}_{t-1}, \dot{C}_{t-1}, \dot{D}_{t-1}, \dot{G}_{t-1}) \\ \dot{C}_t &= j(\dot{Y}_{t-1}, \dot{A}_{t-1}, \dot{B}_{t-1}, \dot{C}_{t-1}, \dot{D}_{t-1}, \dot{G}_{t-1}) \\ \dot{D}_t &= l(\dot{Y}_{t-1}, \dot{A}_{t-1}, \dot{B}_{t-1}, \dot{C}_{t-1}, \dot{D}_{t-1}, \dot{G}_{t-1}) \\ \dot{G}_t &= m(\dot{Y}_{t-1}, \dot{A}_{t-1}, \dot{B}_{t-1}, \dot{C}_{t-1}, \dot{D}_{t-1}, \dot{G}_{t-1})\end{aligned}$$

The Jacobian matrix of this system of equations is:

$$J = \begin{bmatrix} \frac{\partial f}{\partial \dot{Y}_{t-1}} & \frac{\partial f}{\partial \dot{A}_{t-1}} & \frac{\partial f}{\partial \dot{B}_{t-1}} & \frac{\partial f}{\partial \dot{C}_{t-1}} & \frac{\partial f}{\partial \dot{D}_{t-1}} & \frac{\partial f}{\partial \dot{G}_{t-1}} \\ \frac{\partial g}{\partial \dot{Y}_{t-1}} & \frac{\partial g}{\partial \dot{A}_{t-1}} & \frac{\partial g}{\partial \dot{B}_{t-1}} & \frac{\partial g}{\partial \dot{C}_{t-1}} & \frac{\partial g}{\partial \dot{D}_{t-1}} & \frac{\partial g}{\partial \dot{G}_{t-1}} \\ \frac{\partial h}{\partial \dot{Y}_{t-1}} & \frac{\partial h}{\partial \dot{A}_{t-1}} & \frac{\partial h}{\partial \dot{B}_{t-1}} & \frac{\partial h}{\partial \dot{C}_{t-1}} & \frac{\partial h}{\partial \dot{D}_{t-1}} & \frac{\partial h}{\partial \dot{G}_{t-1}} \\ \frac{\partial j}{\partial \dot{Y}_{t-1}} & \frac{\partial j}{\partial \dot{A}_{t-1}} & \frac{\partial j}{\partial \dot{B}_{t-1}} & \frac{\partial j}{\partial \dot{C}_{t-1}} & \frac{\partial j}{\partial \dot{D}_{t-1}} & \frac{\partial j}{\partial \dot{G}_{t-1}} \\ \frac{\partial l}{\partial \dot{Y}_{t-1}} & \frac{\partial l}{\partial \dot{A}_{t-1}} & \frac{\partial l}{\partial \dot{B}_{t-1}} & \frac{\partial l}{\partial \dot{C}_{t-1}} & \frac{\partial l}{\partial \dot{D}_{t-1}} & \frac{\partial l}{\partial \dot{G}_{t-1}} \\ \frac{\partial m}{\partial \dot{Y}_{t-1}} & \frac{\partial m}{\partial \dot{A}_{t-1}} & \frac{\partial m}{\partial \dot{B}_{t-1}} & \frac{\partial m}{\partial \dot{C}_{t-1}} & \frac{\partial m}{\partial \dot{D}_{t-1}} & \frac{\partial m}{\partial \dot{G}_{t-1}} \end{bmatrix}$$
  

$$J = \begin{bmatrix} \frac{\partial f}{\partial \dot{Y}_{t-1}} & \frac{\partial f}{\partial \dot{A}_{t-1}} & s + \frac{k+1}{3}s & 0 & \frac{(k+1)(s-1)}{3} & -\frac{(k+1)s}{3} \\ 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{bmatrix} \quad (4.8)$$

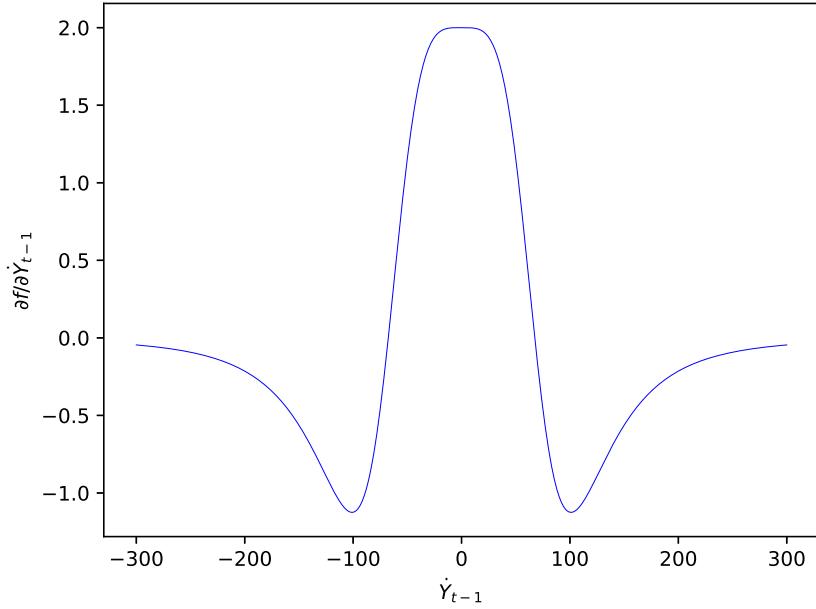


Figure 4.16: Plot of  $\frac{\partial f}{\partial \dot{Y}_{t-1}}$  with respect to  $\dot{Y}_{t-1}$ .

where

$$\begin{aligned}\frac{\partial f}{\partial \dot{Y}_{t-1}} &= \frac{1}{v \left( q + \frac{Y_{t-1}^4}{v^4} \right)} - \frac{4Y_{t-1}^4}{v^5 \left( q + \frac{Y_{t-1}^4}{v^4} \right)} \\ \frac{\partial f}{\partial \dot{A}_{t-1}} &= 1 + \frac{1}{3}(k+1)(1-s) - s + \frac{4A_{t-1}^4}{v^5 \left( q + \frac{A_{t-1}^4}{v^4} \right)^2} - \frac{1}{v \left( q + \frac{A_{t-1}^4}{v^4} \right)}\end{aligned}$$

Figure 4.16 and 4.17 displays the possible values for the variable partial derivatives of the model over a reasonable range of growth rates. All other partial derivatives present in the Jacobian are clearly constant given a set of parameters.

If we think of the our mapping as a transformation in an  $n$ -dimensional space and the Lyapunov exponent as a measure of the compression and expansion of this space, it is clear that we must track the direction of this action as the transformation iterates through time. This is accomplished by evaluating the evolution of an  $n$ -length vector  $v$ , normalized to a length of 1 for simplicity as the direction of this vector is all we are concerned with. This occurs with the 1-dimensional  $v_t$  case as well, by incorporating a 1-dimensional vector of length 1. This is however, equivalent to the scalar 1 so it need not be explicitly included in calculation. The Jacobian  $J$  operates as the higher-dimension analogue to the derivative so by taking the product of the Jacobian and the direction

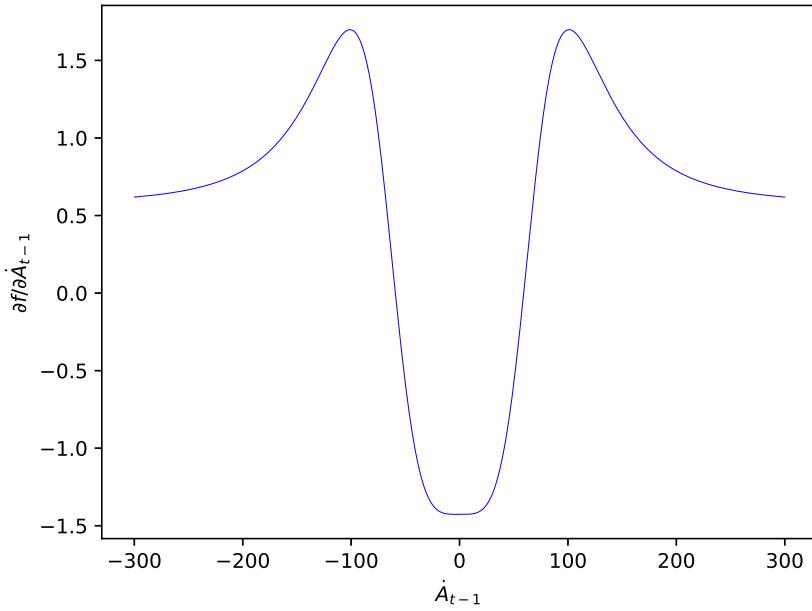


Figure 4.17: Plot of  $\frac{\partial f}{\partial \dot{A}_{t-1}}$  with respect to  $\dot{A}_{t-1} = \dot{Y}_{t-2}$ .

vector, we are able to project our derivatives onto the direction vector. From here, we can solve for the Lyapunov exponent much like in our 1-dimensional case:

$$\lambda(\dot{Y}, \dot{A}, \dot{B}\dot{C}, \dot{D}, \dot{G}) = \lim_{j \rightarrow \infty} \frac{1}{j} \sum_{j=1}^{t=j} \ln |J_t \cdot v_t| \quad (4.9)$$

With a 1-dimensional mapping,  $v_t = v = 1$  and  $J_t = f'(x_t)$ . It is clear how the Jacobian evolves over time as 4.3 is dependent on the state of the economy. The direction vector must also be updated, this is accomplished by normalizing the length of the projection:

$$v_{t+1} = \frac{J_t \cdot v_t}{|J_t \cdot v_t|} \quad (4.10)$$

A consequence of this is that the Lyapunov exponent is now both dependent on the initial condition and the choice of initial direction vector.<sup>12,19</sup> As stated in the introduction, the initial condition provided it not reside in an unstable periodic cycle; moreover, the choice of initial vector is of little consequence as it quickly updates to the correct elongation direction regardless of initial choice. As stated for the 1-dimensional case, it is typically impractical to solve for the Lyapunov exponent analytically; however, we can approximate it by iterating the model for some large  $t$  and treating this value as the limit.

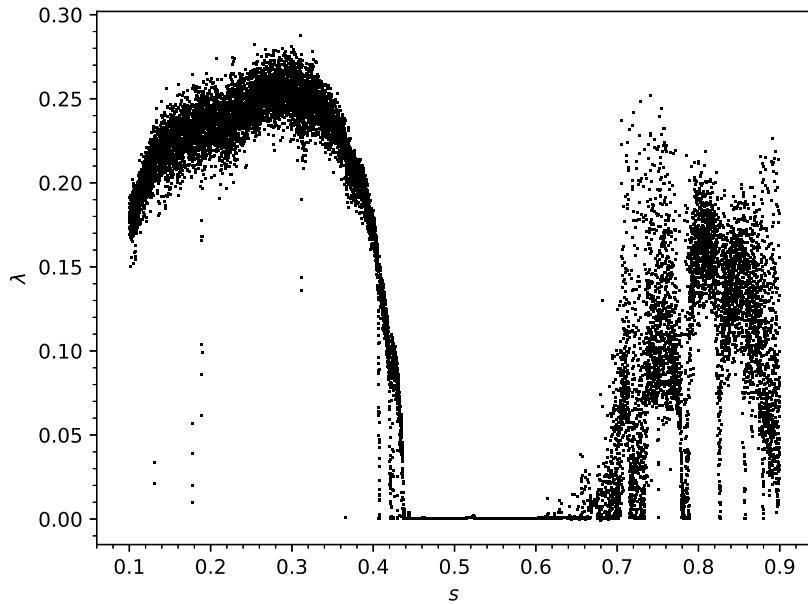


Figure 4.18: Lyapunov plot varying  $s$  between 0 and 1. Other parameters and initial conditions are held constant as described in Figure 4.4.

An important note is that although 1-dimensional models only have 1 Lyapunov exponent,  $n$  dimensional models actually have  $n$  Lyapunov exponents. The method described above only calculates the largest exponent; however, it is typically only necessary to determine this maximal value as the positivity of this exponent is used as the determiner of chaos in the system. A consequence of the bounded behavior of our model is that the negative Lyapunov exponents must "dominate" in order to result in overall compression of the model; otherwise the mapping would result in an explosive model.

Note again that we are calculating only the maximal Lyapunov exponent, thus if the displayed Lyapunov exponent is 0, the other exponents may be 0 or even negative meaning that there can be compression of the space occurring even when  $\lambda = 0$ . Comparing the Lyapunov plot to the bifurcation diagrams for each parameter, we can clearly see that the parameter ranges where the Lyapunov exponent is positive correspond to dense regions in the bifurcation diagram. This implies that parameter choices can result in chaotic behavior as opposed to just cycles of higher periodicity than what was displayed in the bifurcation diagram. The regions of chaotic behavior are unsurprising based on visual observation of the bifurcation diagrams aside from the  $q$  parameter. Figure 4.22 is created identically to Figure 4.8 but by excluding the smallest values for  $q$  allows us to generate a much more accurate picture of the dynamics of the model as  $q$  varies. Here it is obvious

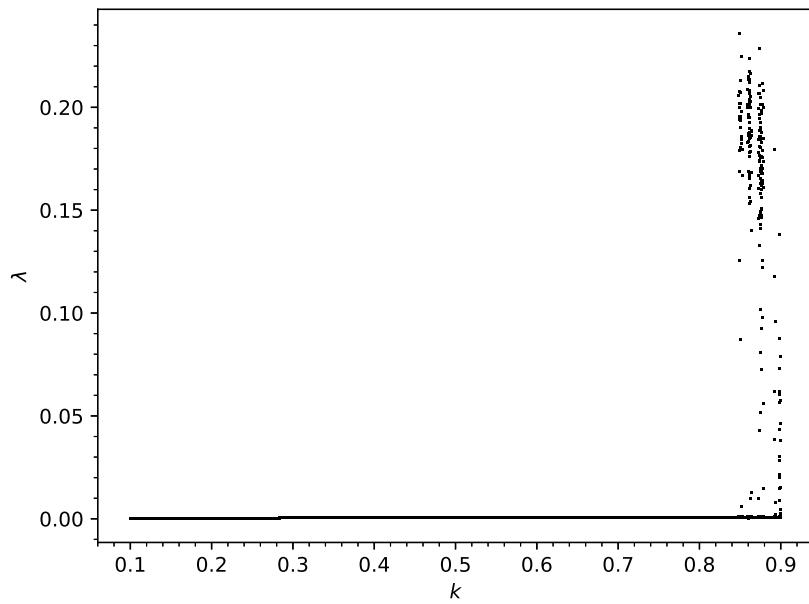


Figure 4.19: Lyapunov plot varying  $k$  between 0 and 1. Other parameters and initial conditions are held constant as described in Figure 4.4

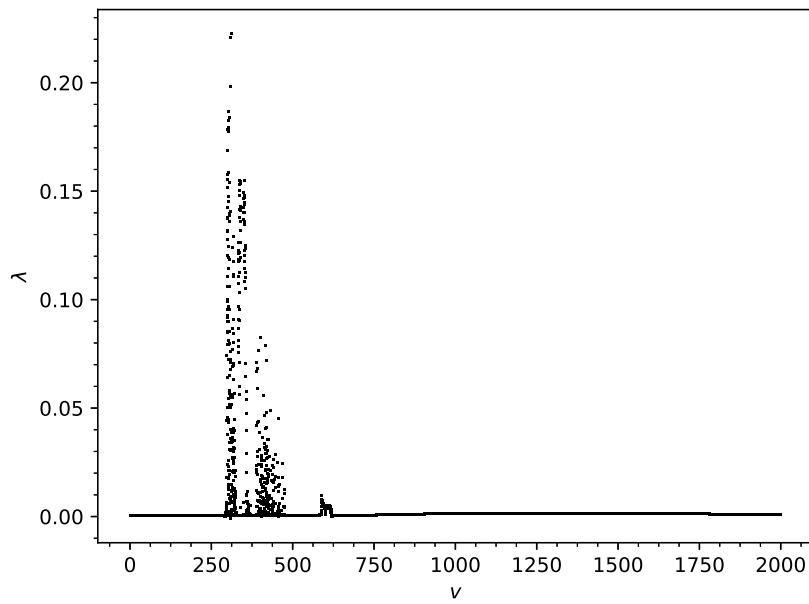


Figure 4.20: Lyapunov plot varying  $v$  between 1 and 2000. Other parameters and initial conditions are held constant as described in Figure 4.4

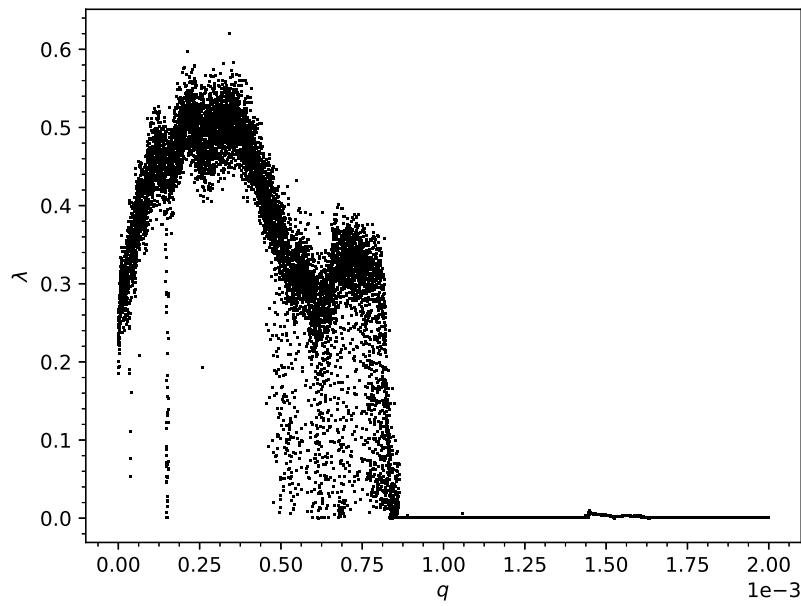


Figure 4.21: Lyapunov plot varying  $q$  between 0 and 0.002. Other parameters and initial conditions are held constant as described in Figure 4.4

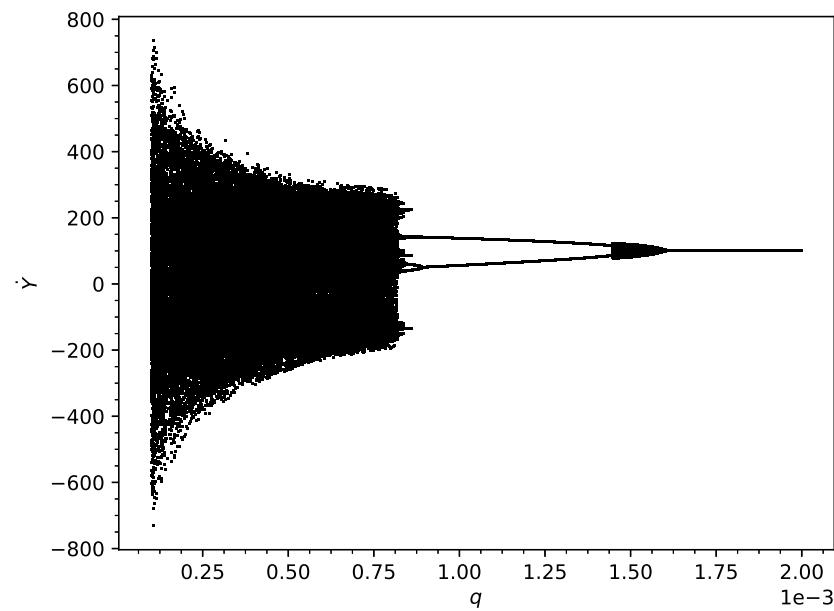


Figure 4.22: Bifurcation diagram varying  $q$  between 0.0001 and 0.002. All other parameters and initial conditions kept as described in Figure 4.4

why the Lyapunov exponent plot indicated the presence of chaos until  $q$  entering a region of stable cycles at  $q > 0.889 \times 10^{-3}$ .

## 5. Conclusion

One of the core debates of economic theory is that of the rationality of agents. Keynesian economics was the dominant school of thought until the 1970s which saw the rise of New Classical economics.<sup>20</sup> Lucas and Sargent were two key spearheads of this movement with their influential paper titled "After Keynesian Macroeconomics", key to their complaints were the lack of microeconomic foundations in Keynesian models.<sup>15</sup> Real business cycle theory, a major branch of New Classical economics believed that business cycles were a rational response to exogenous shocks to the economy and were perfectly efficient. However, this also meant that business cycles would not arise endogenously as firms and consumers would learn over time the behavior of the economy as they, as a whole, acted rationally. Though Keynesianism has since gained in popularity after the 2007 financial crisis, these lessons of New Classical Economics continue to influence the microfoundations of macroeconomic New Keynesian models.

Wegener, et al. directly based their inventory cycle model on that developed by Metzler. However, both of these models required that firms be boundedly rational which, to a New Classical economist, would be an unrealistic assumption not rooted in microeconomic foundations. However, it is appropriate to limit the ability for firms to predict the future if the reason for it can be appropriately explained. Behavioral economics and experimental data has frequently shown a gap between perfectly rational, utilitarian behavior and what is actually practiced by humans;<sup>Smith2006</sup> however, we would expect firms to behave more rationally than individuals. Some models make use of stochastic shocks in order to drive dynamic behavior; these tend to follow a New Classical real business cycle framework as these stochastic processes are the primary drivers of economic cycles. Another possible process is that of chaotic behavior. Under these conditions, it is possible for the economy to reside in a completely deterministic state while still making it impossible for firms to accurately predict future states in the economy . Having the unpredictability of the economy be primarily driven by chaotic processes rather than stochastic ones is of great use from a policy perspective due to the nature of having parameters to influence rather than random behavior, this is discussed in greater detail by Faggini and Parziale.<sup>21</sup>

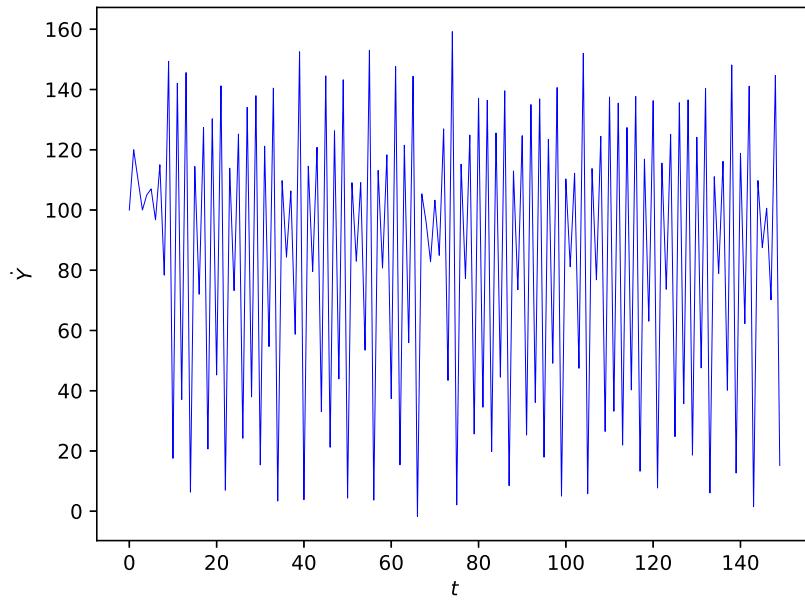


Figure 5.1: Timeseries plot of income growth rate over 150 iterations.  $s = 0.4$ ,  $k = 0.3$ ,  $v = 500$ ,  $q = 0.001$ . Initial values of  $\dot{Y}$  are: 100, 120, 110, 100, 105, 107

The growth model described in Chapter 4 has a basis in Metzler's inventory cycle but also incorporates a multiplier-accelerator mechanism for endogenous investment and consumption and the results of numerical simulations of the model display the possibility of chaos, in addition to other phenomena such as the existence of multiple attractors. Of particular note is the bifurcation diagram and Lyapunov exponent plot of the parameter  $s$ . The marginal propensity to save, under the initial conditions and other parameter choice tested, do not display any stable fixed points. In fact, over the possible range of  $s$ , most possible trajectories are chaotic; one such trajectory is displayed in Figure 5.1. Figure 4.19 also displays chaotic behavior; however, this only occurs when  $k$  is very large and is unlikely to occur in a real scenario. Both  $s$  and  $k$  are the two parameter choices that describe the behavior of the agents of the economy in non-monetary terms. Based on the results of Figure 4.19 and 4.6, variation of this parameter has minor effects on the growth rate of the economy compared to variation of  $s$ . This implies that the choice of inventory proportion, outside of the extreme cases, has a minor impact on the long-run dynamics of the model. This is not to say that the inventory cycle is itself a minor factor of the economy; removal of the inventory cycle changes this model to an unsimplified version of the model presented in Chapter 2. This model features a functionally identical mechanism for consumption with a Robertson lag and although the form of the function

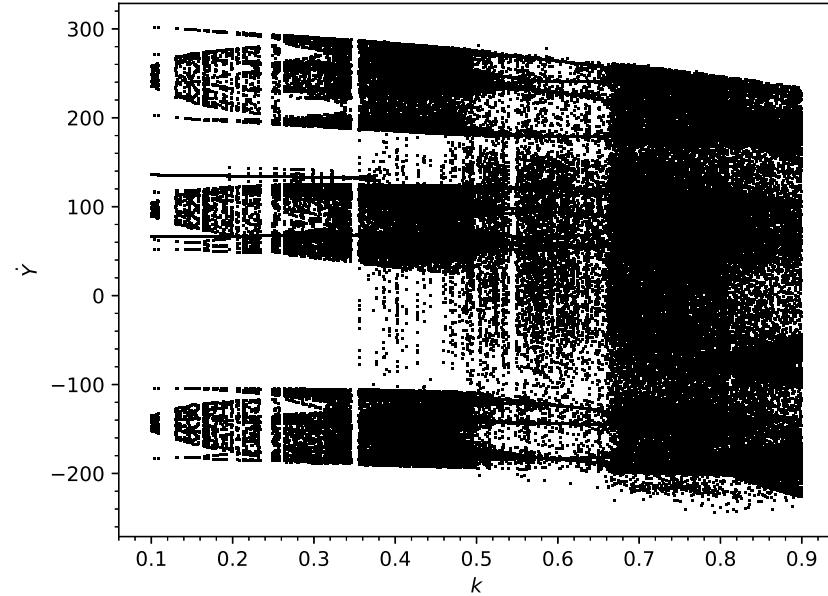


Figure 5.2: Bifurcation diagram varying  $k$  between 0.1 and 0.9. Initial conditions and other parameters are held constant as described in Figure 4.1 except  $s = 0.7$

for endogenous investment differs between the two models, they are qualitatively similar for the reasons mentioned in Chapter 4.

Figure 5.2 and 5.3 displays behavior that is qualitatively distinct from that in Figure 4.6 despite both plots showing the long-run behavior of the model with the same variation in  $k$ . However, by increasing or decreasing  $s$  until it is in the chaotic regime presented in Figure 4.18, we see that varying  $k$  can have a significant effect on the long-run dynamics of growth at even more reasonable values of  $k$ .

$q$  determines the maximum and minimum of the investment function and  $v$  is the primary determinant of the FWHM of the function. Given the initial conditions, we see chaotic behavior when large quantities of investment are allowed by the investment curve, i.e.  $q$  is small. The initial growth values vary between 100 and 120 so it appears that allowing investment to exceed growth in quantity results in instability; however, there do still remain windows of order in the primary chaotic region of  $q$ . Interestingly, the system appears to be relatively ordered for most values of  $v$ ; however, there is a small region where  $250 < v < 750$  that displays chaotic behavior in Figure 4.20. Figure 5.4 displays one such trajectory, of note is how the model cycles between both positive and negative growth regimes but also between high amplitude variation and low amplitude variation. It appears that under this parametrization, high absolute values of income change reduce

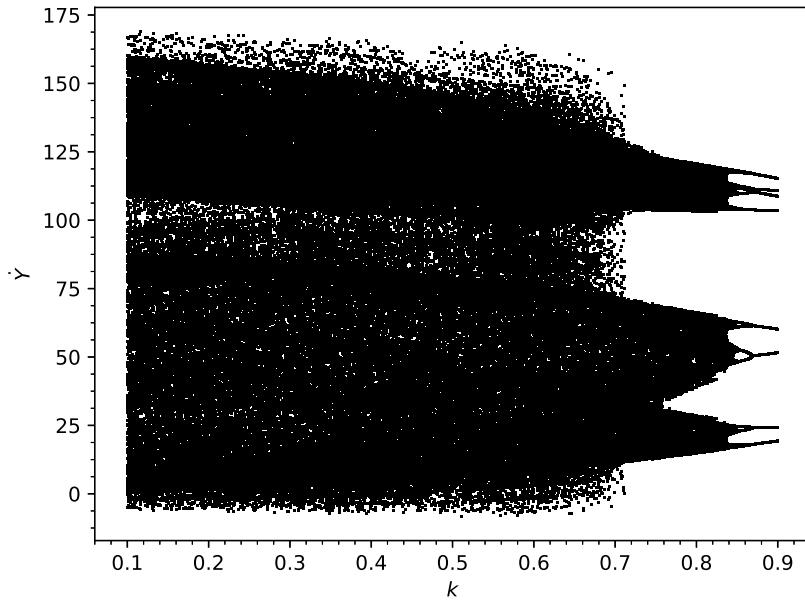


Figure 5.3: Bifurcation diagram varying  $k$  between 0.1 and 0.9. Initial conditions and other parameters are held constant as described in Figure 4.1 except  $s = 0.3$

investment; however, instead of ending in stability, this lower income variance state drives higher levels of investment shifting the model back to a high income variance regime.

$v$  and  $q$  are likely to change depending on the state of the economy, the government's fiscal policy, and the technology available. The investment curve bends back and has a "width" due to the presumption of changes in public investment and taxes in response to the state of the economy and the presence of other limiting factors in income growth. If the government decides to loosen their concern to output variation, the width of the investment curve would increase. The height of the function can change over time as the required level of investment needed to sustain an output level changes can change with improvements in technology or human capital.

Generally speaking, larger values of early initial growth compared to the later values of initial growth results in a lower overall long-run growth. This can be seen in Figure 4.10, 4.11, and 4.12. The relationship is reversed if growth is larger for more recent time periods as can be seen in Figure 4.13, 4.14, and 4.15. Ignoring the chaotic behavior, this model implies that relatively sharp drops in growth rate result in lower growth horizons whereas large spikes in growth rate result in high growth horizons. The non-linear nature of the model and presence of multiple attractors complicates this idea although the trend is still present. This economy is susceptible to external forces; an exogenous but temporary

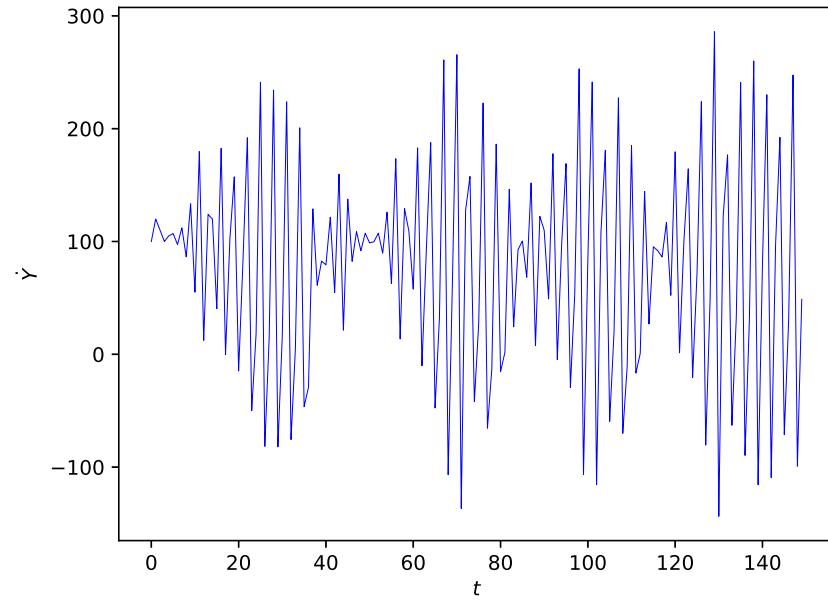


Figure 5.4: Timeseries plot of income growth rate over 150 iterations.  $s = 0.6$ ,  $k = 0.3$ ,  $v = 416$ ,  $q = 0.001$

increase in the growth rate will result in a long-run increase in growth rate. The economy is thus able to use aid in order to accelerate its own growth. Likewise, if an external shock reduces output relative to its usual behavior, this can result in a decreased growth rate in the long-run.

The goal of this model was to incorporate both a Robertson and Lundberg lag and endogenize investment change. This was done in order to provide a mechanism to explain why firms on aggregate are unable to accurately predict future consumption. Many models use stochastic processes in order to rationalize this; however, this model shows the presence of chaotic behavior. Under a chaotic regime, firms would only be able to predict future behavior by knowing the exact state of the economy in the past; however, this is practically impossible as even small differences between the actual initial conditions and estimated initial conditions will result in very different future results by the definition of chaos.

Much work remains to be done on this model in order to better describe the quantity and structure of the multiple attractors and an explanation of the model's response to changing parameters. As more work is done on the mathematics of higher-order non-linear iterated maps, it may become possible to analytically solve for the behavior of the model although this remains unlikely for the near future. More work also needs to be done to

differentiate between quasi-periodic behavior and chaotic behavior in the model, although both are inherently unpredictable, quasi-periodic behavior provides a more consistent structure to long-run behavior while still preventing firms from accurately determining future outcomes.

This model could also be expanded upon in a variety of ways. This economy is closed; however, most modern economies are open to foreign trade to varying degrees. This provides another income channel as well as providing another outlet for investment that does not improve the domestic economy. A more complex investment mechanism could also be incorporated, by explicitly separating public and private investments and removing the x-axis symmetry seen in the current curve. There are many possible ways for firms to predict future consumption, a simple average was used here although a conditional expectation that would allow firms to better use past information and learn from their previous experiences would help alleviate concerns regarding rational behavior by the firms. This however can lead to mathematical complications if firms have infinite possible memory. These additions could improve the realism of the model while still using chaos as the primary driver of chaotic behavior.

## 6. References

- (1) Aubin, D.; Dalmedico, A. D. Writing the history of dynamical systems and chaos: Longue durée and revolution, disciplines and cultures. *Historia Mathematica* **2002**, *29*, 273–339.
- (2) Winfree, A. T. The prehistory of the Belousov-Zhabotinsky oscillator. *Journal of Chemical Education* **1984**, *61*, 661.
- (3) Field, R. J.; Foersterling, H. D. On the oxybromine chemistry rate constants with cerium ions in the Field-Koeroes-Noyes mechanism of the Belousov-Zhabotinskii reaction: the equilibrium  $\text{ch}\{\text{HBrO}_2 + \text{BrO}_3^- + \text{H}^+ \rightleftharpoons 2\text{BrO}_2 + \text{H}_2\text{O}\}$ . *The Journal of Physical Chemistry* **1986**, *90*, 5400–5407.
- (4) Beutel, K. M.; Peacock-López, E. Chemical Oscillations: Two-Variable Chemical Models. **2007**, *12*, 224–235.
- (5) Hudson, J. L.; Tsotsis, T. T. Electrochemical reaction dynamics: a review., 1994.
- (6) Matalon, M. Flame dynamics. *Proceedings of the Combustion Institute* **2009**, *32 I*, 57–82.
- (7) Poincaré, H.; Goroff, D. L., *New Methods of celestial mechanics*; American Institute of Physics: 1993, p 1077.
- (8) Györgyi, L.; Field, R. J. A three-variable model of deterministic chaos in the Belousov-Zhabotinsky reaction. *Nature* **1992**, *355*, 808–810.
- (9) Kavanagh, A. J.; Richards, O. W. The Autocatalytic Growth-Curve. *The American Naturalist* **1934**, *68*, 480.
- (10) Zwanzig, R. Generalized Verhulst laws for population growth. *Proceedings of the National Academy of Sciences of the United States of America* **1973**, *70*, 3048–51.
- (11) May, R. M. Simple mathematical models with very complicated dynamics. *Nature* **1976**, *261*, 459–467.
- (12) Puu, T., *Attractors, Bifurcations, & Chaos*, 2nd Editio; Springer-Verlag: Berlin & Heidelberg, 2003, p 549.

- (13) Samuelson, P. A. Interactions between the Multiplier Analysis and the Principle of Acceleration. *The Review of Economics and Statistics* **1939**, *21*, 75–78.
- (14) Jorgenson, D. W. Capital Theory and Investment Behavior. *American Economic Review* **1963**, *53*, 247–259.
- (15) Lucas, R.; Sargent, T. After Keynesian macroeconomics [conference paper]. *Federal Reserve Bank of Minneapolis Quarterly Review* **1979**, *3*, 1–16.
- (16) Skousen, M. The Perseverance of Paul Samuelson's Economics. *Journal of Economic Perspectives* **1997**, *11*, 137–152.
- (17) Wegener, M.; Westerhoff, F.; Zaklan, G. A Metzlerian business cycle model with nonlinear heterogeneous expectations. *Economic Modelling* **2009**, *26*, 715–720.
- (18) Metzler, L. A. The Nature and Stability of Inventory Cycles. *The Review of Economics and Statistics* **1941**, *23*, 113.
- (19) Medio, A.; Lines, M., *Nonlinear Dynamics: A Primer*; Cambridge University Press: Cambridge ; New York, 2001, p 316.
- (20) *Real Business Cycles: A Reader*, 1st ed.; Hartley, J. E., Hoover, K. D., Salyer, K. D., Eds.; Routledge: New York, NY, 2013, p 669.
- (21) Faggini, M.; Parziale, A. The Failure of Economic Theory. Lessons from Chaos Theory. *Modern Economy* **2012**, *3*, 1–10.

## A. Solving for Change in Income

We can define the growth rate  $\dot{Y}_t$  as the difference in income between contiguous time periods:

$$\dot{Y}_t = Y_{t+1} - Y_t \quad (\text{A.1})$$

Let this dot notation refer to the analogous growth rate of other state variables. This equation can be expanded using Equation 4.6.

$$\dot{Y}_t = (U_{t+1} - U_t) + (I_{t+1} - I_t) + (S_{t+1} - S_t) = \dot{U}_t + \dot{I}_t + \dot{S}_t$$

The change in investment can be trivially solved for as investment is already a function of the change in income.

$$I_t = \frac{\frac{\dot{Y}_{t-2}}{v}}{\left(\frac{\dot{Y}_{t-2}}{v}\right)^4 + q}$$

Thus the change in investment is:

$$\dot{I}_t = \frac{\frac{\dot{Y}_{t-1}}{v}}{\left(\frac{\dot{Y}_{t-1}}{v}\right)^4 + q} - \frac{\frac{\dot{Y}_{t-2}}{v}}{\left(\frac{\dot{Y}_{t-2}}{v}\right)^4 + q} \quad (\text{A.2})$$

As predicted consumption is a function of the change in actual consumption, we must derive a function for the change in consumption in terms of the growth rate.

$$C_{t+1} - C_t = [(1-s)Y_t + sY_{t-1}] - [(1-s)Y_{t-1} + sY_{t-2}]$$

This can be simplified to:

$$\dot{C}_t = (1-s)(Y_t - Y_{t-1}) + s(Y_{t-1} - Y_{t-2}) = (1-s)\dot{Y}_{t-1} + s\dot{Y}_{t-2} \quad (\text{A.3})$$

The change in expected consumption can be given by:

$$U_{t+1} - U_t = \frac{C_t + C_{t-1} + C_{t-2}}{3} - \frac{C_{t-1} + C_{t-2} + C_{t-3}}{3}$$

$$\dot{U}_t = \frac{\dot{C}_{t-1} + \dot{C}_{t-2} + \dot{C}_{t-3}}{3}$$

This can be clearly resolved into a function of income growth rates:

$$\dot{U}_t = \frac{[(1-s)\dot{Y}_{t-2} + s\dot{Y}_{t-3}] + [(1-s)\dot{Y}_{t-3} + s\dot{Y}_{t-4}] + [(1-s)\dot{Y}_{t-4} + s\dot{Y}_{t-5}]}{3}$$

$$\dot{U}_t = \frac{1}{3} \left( (1-s)(\dot{Y}_{t-2} + \dot{Y}_{t-3} + \dot{Y}_{t-4}) + s(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) \right) \quad (\text{A.4})$$

The final step is to solve for inventory production in terms of growth rates. We can substitute our  $S_t$  into our function for  $Q_t$ :

$$Q_t = Q_{t-1} + (kU_t - Q_{t-1} + (U_t - C_t))$$

$$Q_t = kU_t + U_t - C_t$$

This gives an equation for the rate of change:

$$Q_{t+1} - Q_t = [kU_{t+1} + U_{t+1} - C_{t+1}] - [kU_t + U_t - C_t]$$

$$\dot{Q}_t = (k+1)(U_{t+1}) - (k+1)(U_t) - (C_{t+1} - C_t)$$

$$\dot{Q}_t = (k+1)(U_{t+1} - U_t) - (C_{t+1} - C_t)$$

$$\dot{Q}_t = (k+1)(\dot{U}_t) - (\dot{C}_t)$$

Thus giving a function with respect to income growth rate:

$$\dot{Q}_t = \frac{k+1}{3} \left( (1-s)(\dot{Y}_{t-2} + \dot{Y}_{t-3} + \dot{Y}_{t-4}) + s(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) \right) \quad (\text{A.5})$$

$$- (1-s)\dot{Y}_{t-1} - s\dot{Y}_{t-2}$$

The difference in production of goods for inventory is given by:

$$S_{t+1} - S_t = (kU_{t+1} - Q_t) - (kU_t - Q_{t-1})$$

$$\dot{S}_t = k(U_{t+1} - U_t) + Q_{t-1} - Q_t$$

$$\dot{S}_t = k(\dot{U}_t) - (\dot{Q}_{t-1})$$

We can thus substitute our functions for the change in predicted consumption and change

in inventory:

$$\begin{aligned}
 \dot{S}_t &= \frac{k}{3} \left( (1-s)(\dot{Y}_{t-2} + \dot{Y}_{t-3} + \dot{Y}_{t-4}) + s(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) \right) - \\
 &\quad \left[ \frac{k+1}{3} \left( (1-s)(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) + s(\dot{Y}_{t-4} + \dot{Y}_{t-5} + \dot{Y}_{t-6}) \right) \right. \\
 &\quad \left. - (1-s)\dot{Y}_{t-2} - s\dot{Y}_{t-3} \right] \\
 \dot{S}_t &= \frac{k}{3} \left( (1-s)(\dot{Y}_{t-2} + \dot{Y}_{t-3} + \dot{Y}_{t-4}) + s(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) \right) - \\
 &\quad \frac{k+1}{3} \left( (1-s)(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) + s(\dot{Y}_{t-4} + \dot{Y}_{t-5} + \dot{Y}_{t-6}) \right) \\
 &\quad + (1-s)\dot{Y}_{t-2} + s\dot{Y}_{t-3}
 \end{aligned} \tag{A.6}$$

This means the function for the growth rate of the economy can be written as a 6th order, single-variable difference equation:

$$\begin{aligned}
 \dot{Y}_t &= \frac{\frac{\dot{Y}_{t-1}}{v}}{\left(\frac{\dot{Y}_{t-1}}{v}\right)^4 + q} - \frac{\frac{\dot{Y}_{t-2}}{v}}{\left(\frac{\dot{Y}_{t-2}}{v}\right)^4 + q} + \\
 &\quad \frac{1}{3} \left( (1-s)(\dot{Y}_{t-2} + \dot{Y}_{t-3} + \dot{Y}_{t-4}) + s(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) \right) + \\
 &\quad \frac{k}{3} \left( (1-s)(\dot{Y}_{t-2} + \dot{Y}_{t-3} + \dot{Y}_{t-4}) + s(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) \right) - \\
 &\quad \frac{k+1}{3} \left( (1-s)(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) + s(\dot{Y}_{t-4} + \dot{Y}_{t-5} + \dot{Y}_{t-6}) \right) \\
 &\quad + (1-s)\dot{Y}_{t-2} + s\dot{Y}_{t-3} \\
 \dot{Y}_t &= \frac{\frac{\dot{Y}_{t-1}}{v}}{\left(\frac{\dot{Y}_{t-1}}{v}\right)^4 + q} - \frac{\frac{\dot{Y}_{t-2}}{v}}{\left(\frac{\dot{Y}_{t-2}}{v}\right)^4 + q} + \\
 &\quad \frac{k+1}{3} \left( (1-s)(\dot{Y}_{t-2} + \dot{Y}_{t-3} + \dot{Y}_{t-4}) + s(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) \right) - \\
 &\quad \frac{k+1}{3} \left( (1-s)(\dot{Y}_{t-3} + \dot{Y}_{t-4} + \dot{Y}_{t-5}) + s(\dot{Y}_{t-4} + \dot{Y}_{t-5} + \dot{Y}_{t-6}) \right) \\
 &\quad + (1-s)\dot{Y}_{t-2} + s\dot{Y}_{t-3} \\
 \dot{Y}_t &= \frac{\frac{\dot{Y}_{t-1}}{v}}{\left(\frac{\dot{Y}_{t-1}}{v}\right)^4 + q} - \frac{\frac{\dot{Y}_{t-2}}{v}}{\left(\frac{\dot{Y}_{t-2}}{v}\right)^4 + q} + \\
 &\quad \frac{k+1}{3} \left[ (1-s)(\dot{Y}_{t-2} - Y_{t-5}) + s(\dot{Y}_{t-3} - Y_{t-6}) \right] + (1-s)\dot{Y}_{t-2} + s\dot{Y}_{t-3}
 \end{aligned} \tag{A.7}$$

## B. Computational Analysis of the Logistic Map

### B.1 Cobweb Plot

```
import numpy as np
from matplotlib import rc
import matplotlib.pyplot as plt

#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')

#define Function
def logistic(u, x):
    return u * x * (1-x)

#Plots cobweb plot
def cobweb(u, X0, iterations):
    px, py = np.empty((2, iterations+1, 2))
    px[0], py[0] = X0, X0
    for i in range(1, iterations, 2):
        px[i] = px[i-1]
        py[i] = logistic(u, px[i-1])
        px[i+1] = py[i]
        py[i+1] = py[i]

    x = np.linspace(0,50,1000)
    fig = plt.figure(dpi=1000)
    ax = fig.add_subplot(111)
    ax.plot(x, x, c='gray', lw=1, dashes=[6,2])
    ax.plot(px, py, c='blue', linewidth=0.5)
```

```
#Beautify Plot
ax.minorticks_on()
ax.grid(which='minor', alpha=0.5)
ax.grid(which='major', alpha=0.5)
ax.axhline(0, color='k', lw=.5, alpha=0.25)
ax.axvline(0, color='k', lw=.5, alpha=0.25)
ax.set_aspect('equal')
ax.set_xlabel('$X_t$')
ax.set_ylabel('$X_{t+1}$')
ax.set_xlim(0, 1)
ax.set_ylim(0, 1)
cobweb(1.5, 0.1, 1000)
plt.savefig('./manuscript/figures/logistic/fixed_cob.eps', dpi=1200)
cobweb(3.2, 0.799455, 1000)
plt.savefig('./manuscript/figures/logistic/2-cyclic_cob.eps', dpi=1200)
cobweb(3.82842712, 0.5, 1000)
plt.savefig('./manuscript/figures/logistic/3-cyclic_cob.eps', dpi=1200)
cobweb(3.8, 0.1, 1000)
plt.savefig('./manuscript/figures/logistic/chaos_cob.eps', dpi=1200)
```

## B.2 Bifurcation Diagram

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
import math
import statistics
#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')
```

```
def logistic(u, x):
    return u * x * (1-x)
#Define bifurcation plot
n = 1000
```

```
iterations = 1000
last = 100
x = 0.1 * np.ones(n)
u = np.linspace(0.0001, 4.0, n)
fig = plt.figure()
ax = fig.add_subplot(111)
for i in range(iterations):
    x = logistic(u, x)

    if i >=(iterations - last):
        ax.plot(u, x, ',k', alpha=0.25)

#Labelling
ax.minorticks_on()
ax.set_xlabel('$\mu$')
ax.set_ylabel('x')
ax.set_title('Bifurcation Diagram')
#Save and show
plt.savefig('./manuscript/figures/logistic/bifurcation.eps', dpi=1200)

plt.show()
```

### B.3 Lyapunov Plot

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
import math
import statistics
from tqdm import tqdm
#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')

def logistic(r, x):
    return u * x * (1-x)
```

```
#Range of u values and interval between each
n = 10000
u = np.linspace(0.0001, 4.0, n)

#Higher iteration count increases approximation accuracy
iterations = 100000
#Set initial value
x0 = 0.1
x = 0.1 * np.ones(n)

#Lyapunov calculation
lyapunov = np.zeros(n)
fig = plt.figure()
ax = fig.add_subplot(111)

print('Solving for L')
for i in tqdm(range(iterations)):
    x = logistic(u, x)
    #Compute partial sums using log of absolute value of derivative
    lyapunov += np.log(np.abs(u * (1 - 2 * x)))

#Plotting Lyapunov
ax.plot(u[lyapunov < 0],
        lyapunov[lyapunov < 0] / iterations,
        '.k', alpha=.5, ms=.5)
ax.plot(u[lyapunov >= 0],
        lyapunov[lyapunov >= 0] / iterations,
        '.r', alpha=.5, ms=.5)

#Labelling
ax.minorticks_on()
ax.set_xlabel('$\mu$')
ax.set_ylabel('$\lambda$')
ax.set_title('Lyapunov Plot')
ax.set_ylim(-2, 1)
ax.axhline(0, color='k', lw=.5, alpha=.5)

#Save and show
```

```
plt.savefig('./manuscript/figures/logistic/lyapunov.eps', dpi=1200)  
plt.show()
```

## C. Computational Analysis of the Multiplier-Accelerator Model

### C.1 Cobweb Plot

```
import numpy as np
from matplotlib import rc
import matplotlib.pyplot as plt
from tqdm import tqdm

#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')

#Create function for cobweb plot with arguments: function, parameter, and initial value
def plot_cobweb(f, u, x0, nmax=250 ):
    x = np.linspace(-1,1,2000)
    fig = plt.figure()
    ax = fig.add_subplot(111)

    #Plot map
    ax.plot(x, f(x, u), c='black', lw=1)
    #Plot 45 degree line
    ax.plot(x, x, c='gray', lw=1, dashes=[6,2])

    #Iterate function for nmax stpdf
    px, py = np.empty((2, nmax+1, 2))
    px[0], py[0] = x0, x0
    for n in tqdm(range(1, nmax, 2)):
```

```
px[n] = px[n-1]
py[n] = f(px[n-1], u)
px[n+1] = py[n]
py[n+1] = py[n]

#Plot path
ax.plot(px, py, c='blue', linewidth=0.5)

#Beautify Plot
ax.minorticks_on()
ax.grid(which='minor', alpha=0.5)
ax.grid(which='major', alpha=0.5)
ax.set_aspect('equal')
ax.set_xlabel('$\dot{Y}_t$')
ax.axhline(0, color='k', lw=.5, alpha=0.25)
ax.axvline(0, color='k', lw=.5, alpha=0.25)
ax.set_ylabel('$\dot{Y}_{t+1}$')

class AnnotatedFunction:
    def __init__(self, func, latex_label):
        self.func = func
        self.latex_label = latex_label

    def __call__(self, *args, **kwargs):
        return self.func(*args, **kwargs)

#Logistic Function
func = AnnotatedFunction(lambda x, u: u*x-(u+1)*x**3, r'$\mu_Z-(\mu+1)Z^3$')
plot_cobweb(func, 1.5, 0.1)
plt.savefig('./manuscript/figures/sam_hicks/fixed.eps', dpi=1500)
plot_cobweb(func, 2.15, 0.1)
plt.savefig('./manuscript/figures/sam_hicks/2-cyclic.eps', dpi=1500)
plot_cobweb(func, 2.4, 0.1)
plt.savefig('./manuscript/figures/sam_hicks/chaos_contained.eps', dpi=1500)
plot_cobweb(func, 2.6, 0.1)
plt.savefig('./manuscript/figures/sam_hicks/chaos_uncontained.eps', dpi=1500)
```

```
plot_cobweb(func, 2.7, 0.1)
plt.savefig('./manuscript/figures/sam_hicks/unbound_cyclic.eps', dpi=1500)
```

## C.2 Bifurcation Diagram

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
import math
import time
from tqdm import tqdm
import statistics
#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')

def sam_hick(u, x):
    return u*x-(u+1)*x**3
#Define bifurcation plot
def bifur_2way(n, iterations, x0, u_lower, u_upper):
    last = 50
    x = x0 * np.ones(n)
    u = np.linspace(u_lower, u_upper, n)
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    for i in tqdm(range(iterations)):
        x = sam_hick(u, x)
        if i >=(iterations - last):
            ax.plot(u, x, 'k', alpha=0.25)
    y = -x0 * np.ones(n)
    for i in tqdm(range(iterations)):
        y = sam_hick(u, y)
        if i >=(iterations - last):
            ax.plot(u, y, 'b', alpha=0.25)
#Labelling
ax.minorticks_on()
```

```
ax.set_xlabel('$\mu$')
ax.set_ylabel('$\dot{x}$')
ax.set_title('Bifurcation_Diagram')
ax.set_xlim(u_lower, u_upper)
bifur_2way(5000, 10000, 0.1, 0.9, 3.0)
#Save and show
plt.savefig("./manuscript/figures/sam_hicks/bifurcation.eps", dpi=1200)

plt.show()
```

### C.3 Lyapunov Plot

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
import math
import statistics
from tqdm import tqdm
#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')

def sam_hick(u, x):
    return u*x-(u+1)*x**3
#Range of u values and interval between each
n = 50000
u = np.linspace(0.0001, 3.0, n)

#Higher iteration count increases approximation accuracy
iterations = 100000
#Set initial value
x0 = 0.1
x = 0.1 * np.ones(n)

#Lyapunov calculation
lyapunov = np.zeros(n)
```

```
fig = plt.figure(dpi=1200)
ax = fig.add_subplot(111)

for i in tqdm(range(iterations)):
    x = sam_hick(u, x)
    #Compute partial sums using log of absolute value of derivative
    lyapunov += np.log(abs(u-3*(u+1)*x**2))

#Plotting Lyapunov
ax.plot(u[lyapunov < 0],
         lyapunov[lyapunov < 0] / iterations,
         'k', alpha=.25)
ax.plot(u[lyapunov >= 0],
         lyapunov[lyapunov >= 0] / iterations,
         'r', alpha=.25)

#Labelling
ax.minorticks_on()
ax.set_xlabel('$\mu$')
ax.set_ylabel('$\lambda$')
ax.set_title('Lyapunov_Plot')
ax.set_ylim(-2, 1)
ax.axhline(0, color='k', lw=.5, alpha=0.25)

#Save and show
plt.savefig('./manuscript/figures/sam_hicks/lyapunov.eps', dpi=1500)
plt.show()
```

## D. Computational Analysis of the Metzlerian Inventory Cycle Model with Heterogenous Expectation Rules

### D.1 Timeseries Plot

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
import math
import statistics
from tqdm import tqdm
import scipy

#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')

#Define Function
def metzlerian(k, U, Ulag, Clag, Ibar):
    return U + k * U - (1 + k) * Ulag + Clag + Ibar
#Solves for steady state income
def ssincome(b, Ibar):
    return (1/(1-b)) * Ibar
#Solves for steady state consumption
def ssconsumption(b, Ybar):
    return b * Ybar
#Solves for U, goods produced for sale
def saledist(c, f, d, Cbar, Clag):
```

```
Uextrapolate = Clag + c * (Clag - Cbar)
Uregression = Clag + f * (Cbar - Clag)
w = 1 / (1 + d * (Cbar - Clag)**2)
return w * Uextrapolate + (1 - w)*Uregression
#Solves for consumption
def consumption(b, Ylag):
    return b * Ylag
#Solves for S, goods produced for stock
def stock(k, Ulag, Clag, U):
    Qhatlag = k * Ulag
    Qlag = Qhatlag - (Clag - Ulag)
    Qhat = k * U
    return Qhat - Qlag
#Plots timeseries data
def timeseries_plot(Income):
    fig = plt.figure(dpi=1200)
    #Time series of mapping
    ax1 = fig.add_subplot(1, 1, 1)
    ax1.plot(Income, c='blue', linewidth=0.5)
    plt.ylabel('Income')
    plt.xlabel('t')
    plt.savefig('./manuscript/figures/metzlerian_basic/timeseries_income.eps', dpi
               =1200)
    plt.show()
#Solves for timeseries data
def timeseries(b, c, d, f, k, Y0, U0, Ibar, iterations):
    Ybar = ssincome(b, Ibar)
    Cbar = ssconsumption(b, Ybar)
    Income = np.array([Y0]*iterations)
    Sale = np.array([U0]*iterations)
    Consumption = np.ones(iterations)
    for i in tqdm(range(1, iterations)):
        Consumption[i-1] = consumption(b, Income[i-1])
        Sale[i] = saledist(c, f, d, Cbar, Consumption[i-1])
        Income[i] = metzlerian(k, Sale[i], Sale[i-1], Consumption[i-1], Ibar)
    timeseries_plot(Income)
```

```
timeseries(0.75, 0.3, 1.0, 0.1, 0.1, 40.6, 30.3, 10, 150)
```

## D.2 Cobweb Plot

```
import numpy as np
from matplotlib import rc
import matplotlib.pyplot as plt
from tqdm import tqdm

#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')

#define Function
def metzlerian(k, U, Ulag, Clag, Ibar):
    return U + k * U - (1 + k) * Ulag + Clag + Ibar
#Solves for steady state income
def ssincome(b, Ibar):
    return (1/(1-b)) * Ibar
#Solves for steady state consumption
def ssconsumption(b, Ybar):
    return b * Ybar
#Solves for U, goods produced for sale
def saledist(c, f, d, Cbar, Clag):
    Uextrapolate = Clag + c * (Clag - Cbar)
    Uregression = Clag + f * (Cbar - Clag)
    w = 1 / (1 + d * (Cbar - Clag)**2)
    return w * Uextrapolate + (1 - w)*Uregression
#Solves for consumption
def consumption(b, Ylag):
    return b * Ylag
#Solves for S, goods produced for stock
def stock(k, Ulag, Clag, U):
    qhatlag = k * Ulag
```

```
Qlag = Qhatlag - (Clag - Ulag)
Qhat = k * U
#Plots cobweb plot
def cobweb(b, c, d, f, k, Y0, U0, Ibar, iterations):
    Ybar = ssincome(b, Ibar)
    Cbar = ssconsumption(b, Ybar)
    Income = np.array([Y0]*iterations)
    Sale = np.array([U0]*iterations)
    Consumption = np.ones(iterations)
    Stock = np.ones(iterations)
    for i in tqdm(range(1, iterations)):
        Consumption[i-1] = consumption(b, Income[i-1])
        Sale[i] = saledist(c, f, d, Cbar, Consumption[i-1])
        Income[i] = metzlerian(k, Sale[i], Sale[i-1], Consumption[i-1], Ibar)
    px = []
    py = []
    for i in range(iterations-1):
        px = np.append(px, Income[i])
        py = np.append(py, Income[i+1])
    print(px)
    print(py)

x = np.linspace(0,50,1000)
fig = plt.figure(dpi=1000)
ax = fig.add_subplot(111)
ax.plot(x, x, c='gray', lw=1, dashes=[6,2])
ax.plot(px, py, c='blue', linewidth=0.5)
#Beautify Plot
ax.minorticks_on()
ax.grid(which='minor', alpha=0.5)
ax.grid(which='major', alpha=0.5)
ax.axhline(0, color='k', lw=.5, alpha=0.25)
ax.axvline(0, color='k', lw=.5, alpha=0.25)
ax.set_aspect('equal')
ax.set_xlabel('$Y_t$')
ax.set_ylabel('$Y_{t+1}$')
```

```
ax.set_xlim(38.5, 41.0)
ax.set_ylim(38.5, 41.0)
plt.savefig('./manuscript/figures/metzlerian_basic/cobweb.eps')
plt.show()
cobweb(0.75, 0.3, 1.0, 0.1, 0.1, 40.6, 30.3, 10, 150)
```

## D.3 Bifurcation Diagram

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
from tqdm import tqdm
import statistics

#LaTeX Font
rc('text', usetex=True)
rc('font', family='serif')

#define Function
def metzlerian(k, U, Ulag, Clag, Ibar):
    return U + k * U - (1 + k) * Ulag + Clag + Ibar
#Solves for steady state income
def ssincome(b, Ibar):
    return (1/(1-b)) * Ibar
#Solves for steady state consumption
def ssconsumption(b, Ybar):
    return b * Ybar
#Solves for U, goods produced for sale
def saledist(c, f, d, Cbar, Clag):
    Uextrapolate = Clag + c * (Clag - Cbar)
    Uregression = Clag + f * (Cbar - Clag)
    w = 1 / (1 + d * (Cbar - Clag)**2)
    return w * Uextrapolate + (1 - w)*Uregression
#Solves for consumption
def consumption(b, Ylag):
    return b * Ylag
```

```

#Solves for S, goods produced for stock
def stock(k, Ulag, Clag, U):
    Qhatlag = k * Ulag
    Qlag = Qhatlag - (Clag - Ulag)
    Qhat = k * U
    return Qhat - Qlag
#Solves for timeseries data
def timeseries(b, c, d, f, k, Y0, U0, Ibar, iterations):
    Ybar = ssincome(b, Ibar)
    Cbar = ssconsumption(b, Ybar)
    Income = np.array([Y0]*iterations)
    Sale = np.array([U0]*iterations)
    Consumption = np.ones(iterations)
    for i in range(1, iterations):
        Consumption[i-1] = consumption(b, Income[i-1])
        Sale[i] = saledist(c, f, d, Cbar, Consumption[i-1])
        Income[i] = metzlerian(k, Sale[i], Sale[i-1], Consumption[i-1], Ibar)
    return Income

#Bifurcation Diagram varying mpc, b
def bbifurcation(lower, upper, points, c, d, f, k, Y0, U0, Ibar):
    #Plot set up
    last = 50
    iterations = 1000
    fig = plt.figure(dpi=1000)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    b = np.linspace(lower, upper, points)
    xaxis = np.repeat(b, last)
    yaxis = []
    #Solve for each parameter value
    print("b_Bifurcation")
    for j in tqdm(range(points)):
        Income = timeseries(b[j], c, d, f, k, Y0, U0, Ibar, iterations)
        lastvalues = Income[-last:]
        yaxis.extend(lastvalues)

```

```
ax.plot(xaxis, yaxis, ',k', alpha=0.25)
#Labelling
ax.minorticks_on()
ax.set_xlabel('$b$')
ax.set_ylabel('$Y$')

#Save and show
plt.savefig('./manuscript/figures/metzlerian_basic/bbifurcation.eps', dpi=1500)
#Bifurcation Diagram varying expected deviation from equilibrium, c
def cbifurcation(lower, upper, points, b, d, f, k, Y0, U0, Ibar):
    #Plot set up
    last = 50
    iterations = 1000
    fig = plt.figure(dpi=1000)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    c = np.linspace(lower, upper, points)
    xaxis = np.repeat(c, last)
    yaxis = []
    #Solve for each parameter value
    print("c_Bifurcation")
    for j in tqdm(range(points)):
        Income = timeseries(b, c[j], d, f, k, Y0, U0, Ibar, iterations)
        lastvalues = Income[-last:]
        yaxis.extend(lastvalues)
    ax.plot(xaxis, yaxis, ',k', alpha=0.25)
    #Labelling
    ax.minorticks_on()
    ax.set_xlabel('$c$')
    ax.set_ylabel('$Y$')

#Save and show
plt.savefig('./manuscript/figures/metzlerian_basic/cbifurcation.eps', dpi=1500)
#Bifurcation Diagram varying desired level of consumption relative to expected sale of
consumption goods, k
def kbifurcation(lower, upper, points, b, c, d, f, Y0, U0, Ibar):
```

```
#Plot set up
last = 50
iterations = 1000
fig = plt.figure(dpi=1000)
ax = fig.add_subplot(111)
#Generate distribution of parameter
k = np.linspace(lower, upper, points)
xaxis = np.repeat(k, last)
yaxis = []
#Solve for each parameter value
print("k_Bifurcation")
for j in tqdm(range(points)):
    Income = timeseries(b, c, d, f, k[j], Y0, U0, Ibar, iterations)
    lastvalues = Income[-last:]
    yaxis.extend(lastvalues)
    ax.plot(xaxis, yaxis, ',k', alpha=0.25)
#Labelling
ax.minorticks_on()
ax.set_xlabel('$k$')
ax.set_ylabel('$Y$')

#Save and show
plt.savefig('./manuscript/figures/metzlerian_basic/kbifurcation.eps', dpi=1500)

#Bifurcation Diagram varying popularity of regressive expectations, d
def dbifurcation(lower, upper, points, b, c, f, k, Y0, U0, Ibar):
    #Plot set up
    last = 50
    iterations = 1000
    fig = plt.figure(dpi=1000)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    d = np.linspace(lower, upper, points)
    xaxis = np.repeat(d, last)
    yaxis = []
    #Solve for each parameter value
```

```
print("d_Bifurcation")
for j in tqdm(range(points)):
    Income = timeseries(b, c, d[j], f, k, Y0, U0, Ibar, iterations)
    lastvalues = Income[-last:]
    yaxis.extend(lastvalues)
    ax.plot(xaxis, yaxis, ',k', alpha=0.25)
    #Labelling
    ax.minorticks_on()
    ax.set_xlabel('$d$')
    ax.set_ylabel('$Y$')

    #Save and show
    plt.savefig('./manuscript/figures/metzlerian_basic/dbifurcation.eps', dpi=1500)

#Improved efficiency bifurcation diagram varying f
def fbifurcation(lower, upper, points, b, c, d, k, Y0, U0, Ibar):
    #Plot set up
    last = 50
    iterations = 1000
    fig = plt.figure(dpi=1000)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    f = np.linspace(lower, upper, points)
    xaxis = np.repeat(f, last)
    yaxis = []
    #Solve for each parameter value
    print("f_Bifurcation")
    for j in tqdm(range(points)):
        Income = timeseries(b, c, d, f[j], k, Y0, U0, Ibar, iterations)
        lastvalues = Income[-last:]
        yaxis.extend(lastvalues)
        ax.plot(xaxis, yaxis, ',k', alpha=0.25)
        #Labelling
        ax.minorticks_on()
        ax.set_xlabel('$f$')
        ax.set_ylabel('$Y$')
```

```
#Save and show
plt.savefig('./manuscript/figures/metzlerian_basic/fbifurcation.eps', dpi=1500)

bbifurcation(0.6, 0.85, 10000, 0.3, 1.0, 0.1, 0.1, 40.6, 30.5, 10)
cbifurcation(0.1, 0.9, 10000, 0.75, 1.0, 0.1, 0.1, 40.5, 30.5, 10)
kbifurcation(0.1, 0.4, 10000, 0.75, 0.3, 0.1, 0.1, 40.5, 30.5, 10)
dbifurcation(0.01, 2.0, 10000, 0.75, 0.3, 0.1, 0.1, 40.5, 30.5, 10)
fbifurcation(0.1, 0.5, 10000, 0.75, 0.3, 1.0, 0.1, 40.0, 30.5, 10)
```

## E. Computational Analysis of the Metzlerian Inventory Cycle Model with Averaged Expectations, Endogenous Investment, and Consumption Lag

### E.1 Timeseries Plot

```
import numpy as np
import matplotlib.pyplot as plt

#Time series of growth with mean of last 3 prediction
def growth(dYt1, dYt2, dYt3, dYt5, dYt6, s, k, v, q):
    dYt = (dYt1 / v) / ((dYt1 / v)**4 + q) - (dYt2 / v) / ((dYt2 / v)**4 + q) +
        ((k+1)/3) * ((1-s)*(dYt2-dYt5)+s*(dYt3-dYt6)) + (1-s)*dYt2 + s*dYt3
    return dYt

#Plot timeseries
def variableplot(Variable):
    fig = plt.figure(dpi=1200)
    #Time series of mapping
    ax1 = fig.add_subplot(1, 1, 1)
    ax1.plot(Variable, c='blue', linewidth=0.5)
    ax1.set_xlabel('$t$')
    ax1.set_ylabel('$\dot{Y}$')

#Generate time series data and plots
def mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q, iter):
    #Initialize vectors
    dY = np.append([dY0, dY1, dY2, dY3, dY4, dY5], np.zeros(iter-6))
```

```
#Simulate
for t in range(6, iter):
    dY[t] = growth(dY[t-1], dY[t-2], dY[t-3], dY[t-5], dY[t-6], s, k, v, q)
variableplot(dY)
return dY

mapping(100, 120, 110, 100, 105, 107, 0.6, 0.3, 500, 0.001, 150)
plt.savefig('./manuscript/figures/metzlerian_growth/timeseries1.eps', dpi=1200)
mapping(100, 120, 110, 100, 105, 107, 0.7, 0.3, 500, 0.001, 150)
plt.savefig('./manuscript/figures/metzlerian_growth/timeseries2.eps', dpi=1200)
mapping(29, 20 , 15 , 10, 20 ,50, 0.6, 0.3, 500, 0.001, 150)
plt.savefig('./manuscript/figures/metzlerian_growth/timeseries3.eps', dpi=1200)
mapping(100, 120, 110, 100, 105, 107, 0.3, 0.3, 500, 0.001, 150)
plt.savefig('./manuscript/figures/metzlerian_growth/chaotic_timeseries.eps', dpi=1200)
mapping(100, 120, 110, 100, 105, 107, 0.6, 0.3, 416, 0.001, 150)
plt.savefig('./manuscript/figures/metzlerian_growth/chaotic_timeseries2.eps', dpi=1200)
```

## E.2 Investment Curves

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm
from matplotlib import rc
from mpl_toolkits.mplot3d import Axes3D
from sympy import *
from sympy.solvers import solve
from sympy import Symbol
from sympy.plotting import plot3d

#Plots investment curve given p and q
def investment(v, q, lower, upper, points):
    xaxis = np.linspace(lower, upper, points)
    yaxis = np.ones(points)
    for i in range(points):
        yaxis[i] = (xaxis[i] / v) / ((xaxis[i] / v)**4 + q)
    fig = plt.figure(dpi=1200)
```

```
#Time series of mapping
ax1 = fig.add_subplot(1, 1, 1)
ax1.plot(xaxis, yaxis, c='blue', linewidth=0.5)
ax1.set_xlabel('$Y_{t-1}-Y_{t-2}$')
ax1.set_ylabel('$I_t$')

#Solves for maximal investment given p and q
def maximum(v, q):
    return ((3 ** (3 / 4)) / (4 * q ** (3 / 4) ))

#Solves for the full width at half maximum given v, q, and maximal investment using
#sympy symbolic algebra. Results of solve for this function are functionally identical
#to results in Mathematica
def width(v, q):
    max = maximum(v, q)
    half = max / 2
    Y = Symbol('Y')
    solutions = solve( (Y / v) / ((Y / v) ** 4 + q) - half, Y)
    halfwidth = abs(solutions[1]-solutions[0])
    return halfwidth

#Plots maximal investment given a range of v and q
def max_investment(lowerv, upperv, lowerq, upperq, points):
    fig = plt.figure(dpi=1200)
    ax1 = fig.add_subplot(1, 1, 1)
    v = np.linspace(lowerv, upperv, points)
    q = np.linspace(lowerq, upperq, points)
    maximuminv = np.ones(points)
    for i in range(points):
        maximuminv[i] = maximum(v[i], q[i])
    ax1.plot(q, maximuminv, c='blue', linewidth=0.5)
    ax1.minorticks_on()
    ax1.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    ax1.set_xlabel('$q$')
    ax1.set_ylabel('Maximal Investment')
```

```
#Plots the Full Width at Half Max with a 3-dimensional triangular surface plot
def width_investment(lowerv, upperv, lowerq, upperq, points):
    fig = plt.figure(dpi=1200)
    ax = plt.axes(projection='3d')
    v = np.linspace(lowerv, upperv, points)
    q = np.linspace(lowerq, upperq, points)
    v = np.repeat(v, points)
    q = np.tile(q, points)
    Width = np.ones(points*points)
    for i in tqdm(range(points*points)):
        Width[i] = width(v[i], q[i])
    ax.plot_trisurf(v, q, Width, linewidth=0.2, antialiased=True)
    ax.ticklabel_format(style='sci', axis='y', scilimits=(0,0))
    ax.set_xlabel('$v$')
    ax.set_ylabel('$q$')
    ax.set_zlabel('FWHM')
investment(500, 0.001, -300, 300, 10000)
plt.savefig('./manuscript/figures/metzlerian_growth/investment.eps', dpi=1200)
max_investment(0, 2000, 0.0001, 0.002, 10000)
plt.savefig('./manuscript/figures/metzlerian_growth/maxinvestment.eps', dpi=1200)
width_investment(1, 1000, 0.001, 0.002, 50)
plt.savefig('./manuscript/figures/metzlerian_growth/widhtinvestment.eps', dpi=1200)
```

### E.3 Bifurcation Diagram Varying Parameters

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm

#Solves for growth in time t given past 6 time period growth rate
def growth(dYt1, dYt2, dYt3, dYt5, dYt6, s, k, v, q):
    dYt = (dYt1 / v) / ((dYt1 / v)**4 + q) - (dYt2 / v) / ((dYt2 / v)**4 + q) +
          ((k+1)/3) * ((1-s)*(dYt2-dYt5)+s*(dYt3-dYt6)) + (1-s)*dYt2 + s*dYt3
    return dYt

#Creates a vector of growth rate given parameters
```

```
def mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q, iter):
    #Initialize vectors
    dY = np.append([dY0, dY1, dY2, dY3, dY4, dY5], np.zeros(iter-6))
    #Simulate
    for t in range(6, iter):
        dY[t] = growth(dY[t-1], dY[t-2], dY[t-3], dY[t-5], dY[t-6], s, k, v, q,)
    return dY

# Solves for last 20 values of growth as a single parameter is changed
# All functions below vary a single parameter between 'lower' and 'upper'
def sbifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, k, v, q):
    #Plot set up
    last = 30
    iterations = 2000
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    s = np.linspace(lower, upper, points)
    xaxis = np.repeat(s, last)
    yaxis = []
    #Solve for each parameter value
    print("s_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s[j], k, v, q, iterations)
        for m in range(iterations):
            if m >= (iterations-last):
                yaxis = np.append(yaxis, Income[m])
    ax.plot(xaxis, yaxis, ',k', alpha=0.25)
    #Labelling
    ax.minorticks_on()
    ax.set_xlabel('$s$')
    ax.set_ylabel('$\dot{Y}$')
    #Save and show

def kbifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, v, q):
    #Plot set up
```

```
last = 20
iterations = 1000
fig = plt.figure(dpi=1200)
ax = fig.add_subplot(111)
#Generate distribution of parameter
k = np.linspace(lower, upper, points)
xaxis = np.repeat(k, last)
yaxis = []
#Solve for each parameter value
print("k_Bifurcation")
for j in tqdm(range(points)):
    Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k[j], v, q, iterations)
    for m in range(iterations):
        if m >= (iterations - last):
            yaxis = np.append(yaxis, Income[m])
    ax.plot(xaxis, yaxis, ',k', alpha=0.25)
#Labelling
ax.minorticks_on()
ax.set_xlabel('$k$')
ax.set_ylabel('$\dot{Y}$')
#Save and show

def qbifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, k, v):
    #Plot set up
    last = 20
    iterations = 1000
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    q = np.linspace(lower, upper, points)
    xaxis = np.repeat(q, last)
    yaxis = []
    #Solve for each parameter value
    print("q_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q[j], iterations)
```

```
for m in range(iterations):
    if m >= (iterations - last):
        yaxis = np.append(yaxis, Income[m])
    ax.plot(xaxis, yaxis, ',k', alpha=0.25)
#Labelling
ax.minorticks_on()
plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
ax.set_xlabel('q')
ax.set_ylabel('dot_Y')
#Save and show

def vbifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, k, q):
    #Plot set up
    last = 20
    iterations = 1000
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    v = np.linspace(lower, upper, points)
    xaxis = np.repeat(v, last)
    yaxis = []
    #Solve for each parameter value
    print("v_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v[j], q, iterations)
        for m in range(iterations):
            if m >= (iterations - last):
                yaxis = np.append(yaxis, Income[m])
            ax.plot(xaxis, yaxis, ',k', alpha=0.25)
        #Labelling
        ax.minorticks_on()
        ax.set_xlabel('v')
        ax.set_ylabel('dot_Y')

sbifurcation(0.1, 0.9, 10000, 100, 120, 110, 100, 105, 107, 0.3, 500, 0.001)
```

```
plt.savefig('./manuscript/figures/metzlerian_growth/sbifurcation.eps', dpi=1200)
kbifurcation(0.1, 0.9, 10000, 100, 120, 110, 100, 105, 107, 0.6, 500, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/kbifurcation.eps', dpi=1200)
kbifurcation(0.1, 0.9, 10000, 100, 120, 110, 100, 105, 107, 0.7, 500, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/kbifurcation2.eps', dpi=1200)
kbifurcation(0.1, 0.9, 10000, 100, 120, 110, 100, 105, 107, 0.3, 500, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/kbifurcation3.eps', dpi=1200)
vbifurcation(1, 2000, 10000, 100, 120, 110, 100, 105, 107, 0.6, 0.3, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/vbifurcation.eps', dpi=1200)
qbifurcation(0, 0.002, 10000, 100, 120, 110, 100, 105, 107, 0.6, 0.3, 500)
plt.savefig('./manuscript/figures/metzlerian_growth/qbifurcation.eps', dpi=1200)
qbifurcation(0.0001, 0.002, 10000, 100, 120, 110, 100, 105, 107, 0.6, 0.3, 500)
plt.savefig('./manuscript/figures/metzlerian_growth/q2bifurcation.eps', dpi=1200)
```

## E.4 Bifurcation Diagram Varying Initial Conditions

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm

#Time series with mean of last 3 prediction
def growth(dYt1, dYt2, dYt3, dYt5, dYt6, s, k, v, q):
    dYt = (dYt1 / v) / ((dYt1 / v)**4 + q) - (dYt2 / v) / ((dYt2 / v)**4 + q) +
        ((k+1)/3) * ((1-s)*(dYt2-dYt5)+s*(dYt3-dYt6)) + (1-s)*dYt2 + s*dYt3
    return dY

#Creates mapping for growth model based on growth function
def mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q, iter):
    #Initialize vectors
    dY = np.append([dY0, dY1, dY2, dY3, dY4, dY5], np.zeros(iter-6))
    #Simulate
    for t in range(6, iter):
        dY[t] = growth(dY[t-1], dY[t-2], dY[t-3], dY[t-5], dY[t-6], s, k, v, q)
    return dY

# Solves for last 20 values of growth as a single parameter is changed
```

```
# All functions below vary a single parameter between 'lower' and 'upper'
def y0bifurcation(lower, upper, points, dY1, dY2, dY3, dY4, dY5, s, k, v, q):
    #Plot set up
    last = 20
    iterations = 1000
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    Y0 = np.linspace(lower, upper, points)
    xaxis = np.repeat(Y0, last)
    yaxis = []
    #Solve for each parameter value
    print("$\dot{Y}_0$-Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(Y0[j], dY1, dY2, dY3, dY4, dY5, s, k, v, q, iterations)
        lastvalues = Income[-last:]
        yaxis.extend(lastvalues)
        ax.plot(xaxis, yaxis, ',k', alpha=0.25)
    #Labelling
    ax.minorticks_on()
    ax.set_xlabel('$\dot{Y}_0$')
    ax.set_ylabel('$\dot{Y}$')

def y1bifurcation(lower, upper, points, dY0, dY2, dY3, dY4, dY5, s, k, v, q):
    #Plot set up
    last = 20
    iterations = 1000
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    Y1 = np.linspace(lower, upper, points)
    xaxis = np.repeat(Y1, last)
    yaxis = []
    #Solve for each parameter value
    print("$\dot{Y}_1$-Bifurcation")
    for j in tqdm(range(points)):
```

```
Income = mapping(dY0, Y1[j], dY2, dY3, dY4, dY5, s, k, v, q, iterations)
lastvalues = Income[-last:]
yaxis.extend(lastvalues)
ax.plot(xaxis, yaxis, ',k', alpha=0.25)
#Labelling
ax.minorticks_on()
ax.set_xlabel('$\dot{Y}_1$')
ax.set_ylabel('$\dot{Y}$')

def y2bifurcation(lower, upper, points, dY0, dY1, dY3, dY4, dY5, s, k, v, q):
    #Plot set up
    last = 20
    iterations = 1000
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    Y2 = np.linspace(lower, upper, points)
    xaxis = np.repeat(Y2, last)
    yaxis = []
    #Solve for each parameter value
    print("$\dot{Y}_2$ Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, Y2[j], dY3, dY4, dY5, s, k, v, q, iterations)
        lastvalues = Income[-last:]
        yaxis.extend(lastvalues)
        ax.plot(xaxis, yaxis, ',k', alpha=0.25)
        #Labelling
        ax.minorticks_on()
        ax.set_xlabel('$\dot{Y}_2$')
        ax.set_ylabel('$\dot{Y}$')

def y3bifurcation(lower, upper, points, dY0, dY1, dY2, dY4, dY5, s, k, v, q):
    #Plot set up
    last = 20
    iterations = 1000
    fig = plt.figure(dpi=1200)
```

```
ax = fig.add_subplot(111)
#Generate distribution of parameter
Y3 = np.linspace(lower, upper, points)
xaxis = np.repeat(Y3, last)
yaxis = []
#Solve for each parameter value
print("$\dot{Y}_3$ Bifurcation")
for j in tqdm(range(points)):
    Income = mapping(dY0, dY1, dY2, Y3[j], dY4, dY5, s, k, v, q, iterations)
    lastvalues = Income[-last:]
    yaxis.extend(lastvalues)
ax.plot(xaxis, yaxis, ',k', alpha=0.25)
#Labelling
ax.minorticks_on()
ax.set_xlabel('$\dot{Y}_3$')
ax.set_ylabel('$\dot{Y}$')

def y4bifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY5, s, k, v, q):
    #Plot set up
    last = 20
    iterations = 1000
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    Y4 = np.linspace(lower, upper, points)
    xaxis = np.repeat(Y4, last)
    yaxis = []
    #Solve for each parameter value
    print("$\dot{Y}_4$ Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, Y4[j], dY5, s, k, v, q, iterations)
        lastvalues = Income[-last:]
        yaxis.extend(lastvalues)
    ax.plot(xaxis, yaxis, ',k', alpha=0.25)
    #Labelling
    ax.minorticks_on()
```

```
ax.set_xlabel('$\dot{Y}_4$')
ax.set_ylabel('$\dot{Y}$')

def y5bifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, s, k, v, q):
    #Plot set up
    last = 20
    iterations = 1000
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    Y5 = np.linspace(lower, upper, points)
    xaxis = np.repeat(Y5, last)
    yaxis = []
    #Solve for each parameter value
    print("$\dot{Y}_5$..Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, dY4, Y5[j], s, k, v, q, iterations)
        lastvalues = Income[-last:]
        yaxis.extend(lastvalues)
        ax.plot(xaxis, yaxis, 'k', alpha=0.25)
    #Labelling
    ax.minorticks_on()
    ax.set_xlabel('$\dot{Y}_5$')
    ax.set_ylabel('$\dot{Y}$')

y0bifurcation(-400, 400, 10000, 120, 110, 100, 105, 107, 0.6, 0.3, 500, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/y0bifurcation.eps', dpi=1200)
y1bifurcation(-400, 400, 10000, 100, 110, 100, 105, 107, 0.6, 0.3, 500, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/y1bifurcation.eps', dpi=1200)
y2bifurcation(-400, 400, 10000, 100, 120, 100, 105, 107, 0.6, 0.3, 500, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/y2bifurcation.eps', dpi=1200)
y3bifurcation(-400, 400, 10000, 100, 120, 110, 105, 107, 0.6, 0.3, 500, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/y3bifurcation.eps', dpi=1200)
y4bifurcation(-400, 400, 10000, 100, 120, 110, 100, 107, 0.6, 0.3, 500, 0.001)
plt.savefig('./manuscript/figures/metzlerian_growth/y4bifurcation.eps', dpi=1200)
y5bifurcation(-400, 400, 10000, 100, 120, 110, 100, 105, 0.6, 0.3, 500, 0.001)
```

```
plt.savefig('./manuscript/figures/metzlerian_growth/y5bifurcation.eps', dpi=1200)
```

## E.5 Bifurcation Analyzer

```
import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm

#Create growth formula
#Time series with mean of last 3 prediction
def growth(dYt1, dYt2, dYt3, dYt5, dYt6, s, k, v, q):
    dYt = (dYt1 / v) / ((dYt1 / v)**4 + q) - (dYt2 / v) / ((dYt2 / v)**4 + q) +
          ((k+1)/3) * ((1-s)*(dYt2-dYt5)+s*(dYt3-dYt6)) + (1-s)*dYt2 + s*dYt3
    return dYt

#Creates mapping for growth model based on growth function
def mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q, iter):
    #Initialize vectors
    dY = np.append([dY0, dY1, dY2, dY3, dY4, dY5], np.zeros(iter-6))
    #Simulate
    for t in range(6, iter):
        dY[t] = growth(dY[t-1], dY[t-2], dY[t-3], dY[t-5], dY[t-6], s, k, v, q)
    return dY

#Creates bifurcation data
#Searches plot given a starting point
#l designates a search from left to right
#r designates a search from right to left
#period designates periodicity of cycle of starting point
#precis determines level of precision required for deviation of cycles

def lbifurana(lower, upper, points, Yaxis, lsearch, period, precis):
    last = 30
    param = np.linspace(lower, upper, points)
    Xaxis = np.repeat(param, last)
    bifurfound = False
```

```
i = np.nonzero(Xaxis >= lsearch)
i = i[0][0]
while bifurfound == False:
    ydat = [Yaxis[i]]
    for j in range(i+1, i+last):
        ydat = np.append(ydat, Yaxis[j])
    count = len(set(np.round(ydat, precis)))
    if count != period:
        print("Bifurcation_Point_Found_Searching")
        bifurfound = True
        print("Period_Shift_from")
        print(period)
        print("Bifurcation_Point_at")
        print(Xaxis[i])
        print("Points")
        print(set(np.round(ydat, precis)))
    else:
        i += last
results.write(f'Bifurcation_Point_found_starting_search_from_{lsearch},_left_to_
right_from_{period}_cycle.\n')
results.write(f'Bifurcation_Point_at_{Xaxis[i]}.\\n')
results.write(f'Long-run_values_at_bifurcation:{set(np.round(ydat,_precis))};_
Value_Count_{len(set(np.round(ydat,_precis)))}.\n')
results.write(f'Analysis_done_rounding_to_{precis}_decimal_places.\n\\n')

def rbifurana(lower, upper, points, Yaxis, usearch, period, precis):
    last = 30
    param = np.linspace(lower, upper, points)
    Xaxis = np.repeat(param, last)
    bifurfound = False
    i = np.nonzero(Xaxis >= usearch)
    i = i[0][0]
    while bifurfound == False:
        ydat = [Yaxis[i]]
        for j in range(i+1, i+last):
            ydat = np.append(ydat, Yaxis[j])
```

```
count = len(set(np.round(ydat, precis)))
if count != period:
    print("Bifurcation_Point_Found")
    bifurfound = True
    print("Period_Shift_from")
    print(period)
    print("Bifurcation_Point_at")
    print(Xaxis[i])
    print("Points")
    print(set(np.round(ydat, precis)))
else:
    i -= last
results.write(f"Bifurcation_Point_found_starting_search_from_{usearch},_right_to_"
             f"left_{into}_{period}_cycle.\n")
results.write(f"Bifurcation_Point_Found_{into}_{period}_cycle.\n")
results.write(f"Bifurcation_Point_at_{Xaxis[i]}.\\n")
results.write(f"Long-run_values_at_bifurcation_{set(np.round(ydat,_precis))};_"
             f"Value_Count_{len(set(np.round(ydat,_precis)))}.\n")
results.write(f"Analysis_done_rounding_to_{precis}_decimal_places.\n\\n")

def sbifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, k, v, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f"Initial_conditions_of_s_bifurcation:_lower={lower},_upper={upper},_"
                 f"points={points},_dY0={dY0},_dY1={dY1},_dY2={dY2},_dY3={dY3},_dY4"
                 f"={dY4},_dY5={dY5},_k={k},_v={v},_q={q}\\n\\n")
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    s = np.linspace(lower, upper, points)
    Yaxis = []
    #Solve for each parameter value
    print("s_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s[j], k, v, q, iterations)
        lastvalues = Income[-last:]
```

```
Yaxis.extend(lastvalues)
return(Yaxis)

def kbifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, v, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f'Initial_conditions_of_k_bifurcation: lower={lower}, upper={upper},
                  points={points}, dY0={dY0}, dY1={dY1}, dY2={dY2}, dY3={dY3}, dY4
                  ={dY4}, dY5={dY5}, s={s}, v={v}, q={q}\n\n')
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    k = np.linspace(lower, upper, points)
    Yaxis = []
    #Solve for each parameter value
    print("k_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k[j], v, q, iterations)
        lastvalues = Income[-last:]
        Yaxis.extend(lastvalues)
    return(Yaxis)

def vbifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, k, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f'Initial_conditions_of_v_bifurcation: lower={lower}, upper={upper},
                  points={points}, dY0={dY0}, dY1={dY1}, dY2={dY2}, dY3={dY3}, dY4
                  ={dY4}, dY5={dY5}, s={s}, k={k}, q={q}\n\n')
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    v = np.linspace(lower, upper, points)
    Yaxis = []
    #Solve for each parameter value
    print("v_Bifurcation")
    for j in tqdm(range(points)):
```

```
Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v[j], q, iterations)
lastvalues = Income[-last:]
Yaxis.extend(lastvalues)
return(Yaxis)

def qbifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, k, v):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f'Initial_conditions_of_q_bifurcation: lower={lower}, upper={upper},
                  points={points}, dY0={dY0}, dY1={dY1}, dY2={dY2}, dY3={dY3}, dY4
                  ={dY4}, dY5={dY5}, s={s}, k={k}, v={v}\n{n}\n')

    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    q = np.linspace(lower, upper, points)
    Yaxis = []
    #Solve for each parameter value
    print("q_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q[j], iterations)
        lastvalues = Income[-last:]
        Yaxis.extend(lastvalues)
    return(Yaxis)

#Saves results in a .txt file and appends further data to bottom of file
#Precision typically 0 indicating only whole number changes in growth are appreciable
changes in periodicity
results = open('./computational/metzlerian_growth/bifur-param_results.txt', 'a+')
Yaxis = sbifurcation(0.1, 0.9, 50000, 100, 120, 110, 100, 105, 107, 0.3, 500, 0.001)
lbifurana(0.65, 0.9, 50000, Yaxis, 0.53, 2, 0)
rbifurana(0.65, 0.9, 50000, Yaxis, 0.53, 2, 0)
Yaxis = kbifurcation(0.1, 0.9, 50000, 100, 120, 110, 100, 105, 107, 0.6, 500, 0.001)
lbifurana(0.1, 0.9, 50000, Yaxis, 0.1, 2, 0)
rbifurana(0.1, 0.9, 50000, Yaxis, 0.9, 2, 0)
Yaxis = vbifurcation(1, 2000, 50000, 100, 120, 110, 100, 105, 107, 0.6, 0.3, 0.001)
lbifurana(1, 2000, 50000, Yaxis, 1, 1, 0)
rbifurana(1, 2000, 50000, Yaxis, 1000, 1, 0)
```

```

lbifurana(1, 2000, 50000, Yaxis, 525, 2, 0)
rbifurana(1, 2000, 50000, Yaxis, 525, 2, 0)
Yaxis = qbifurcation(0, 0.002, 50000, 100, 120, 110, 100, 105, 107, 0.6, 0.3, 500)
rbifurana(0, 0.002, 50000, Yaxis, 0.002, 1, 0)
rbifurana(0, 0.002, 50000, Yaxis, 0.001, 2, 0)
lbifurana(0, 0.002, 50000, Yaxis, 0.001, 2, 0)
results.close()

import numpy as np
import matplotlib.pyplot as plt
from tqdm import tqdm

#Create growth formula
#Time series with mean of last 3 prediction
def growth(dYt1, dYt2, dYt3, dYt5, dYt6, s, k, v, q):
    dYt = (dYt1 / v) / ((dYt1 / v)**4 + q) - (dYt2 / v) / ((dYt2 / v)**4 + q) +
        ((k+1)/3) * ((1-s)*(dYt2-dYt5)+s*(dYt3-dYt6)) + (1-s)*dYt2 + s*dYt3
    return dY

#Creates mapping for growth model based on growth function
def mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q, iter):
    #Initialize vectors
    dY = np.append([dY0, dY1, dY2, dY3, dY4, dY5], np.zeros(iter-6))
    #Simulate
    for t in range(6, iter):
        dY[t] = growth(dY[t-1], dY[t-2], dY[t-3], dY[t-5], dY[t-6], s, k, v, q)
    return dY

#Creates bifurcation data
#Searches plot given a starting point
#l designates a search from left to right
#r designates a search from right to left
#period designates periodicity of cycle of starting point
#precis determines level of precision required for deviation of cycles

def lbifurana(lower, upper, points, Yaxis, lsearch, period, precis):
    last = 30

```

```
param = np.linspace(lower, upper, points)
Xaxis = np.repeat(param, last)
bifurfound = False
i = np.nonzero(Xaxis >= lsearch)
i = i[0][0]
while bifurfound == False:
    ydat = [Yaxis[i]]
    for j in range(i+1, i+last):
        ydat = np.append(ydat, Yaxis[j])
    count = len(set(np.round(ydat, precis)))
    if count != period:
        print("Bifurcation_Point_Found_Searching")
        bifurfound = True
        print("Period_Shift_from")
        print(period)
        print("Bifurcation_Point_at")
        print(Xaxis[i])
        print("Points")
        print(set(np.round(ydat, precis)))
    else:
        i += last
results.write(f"Bifurcation_Point_found_starting_search_from_{lsearch},_left_to_
right_from_{period}_cycle.\n")
results.write(f"Bifurcation_Point_at_{Xaxis[i]}.\\n")
results.write(f"Long-run_values_at_bifurcation:{set(np.round(ydat,-precis))};_
Value_Count={len(set(np.round(ydat,-precis)))}.\n")
results.write(f"Analysis_done_rounding_to_{precis}_decimal_places.\n\\n")

def rbifurana(lower, upper, points, Yaxis, usearch, period, precis):
    last = 30
    param = np.linspace(lower, upper, points)
    Xaxis = np.repeat(param, last)
    bifurfound = False
    i = np.nonzero(Xaxis >= usearch)
    i = i[0][0]
    while bifurfound == False:
```

```
ydat = [Yaxis[i]]
for j in range(i+1, i+last):
    ydat = np.append(ydat, Yaxis[j])
count = len(set(np.round(ydat, precis)))
if count != period:
    print("Bifurcation_Point_Found")
    bifurfound = True
    print("Period_Shift_from")
    print(period)
    print("Bifurcation_Point_at")
    print(Xaxis[i])
    print("Points")
    print(set(np.round(ydat, precis)))
else:
    i -= last
results.write(f'Bifurcation_Point_found_starting_search_from_{usearch},_right_to_'
             f'left_into_{period}_cycle.\n')
results.write(f'Bifurcation_Point_Found_into_{period}_cycle.\n')
results.write(f'Bifurcation_Point_at_{Xaxis[i]}.n')
results.write(f'Long-run_values_at_bifurcation:{set(np.round(ydat,_precis))};_'
             f'Value_Count={len(set(np.round(ydat,_precis)))}.\n')
results.write(f'Analysis_done_rounding_to_{precis}_decimal_places.\n\n')

def y0bifurcation(lower, upper, points, dY1, dY2, dY3, dY4, dY5, s, k, v, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f'Initial_conditions_of.$\dot{Y}_0$.bifurcation:_lower={lower},_upper'
                 f'={upper},_points={points},_dY1={dY1},_dY2={dY2},_dY3={dY3},_dY4={'
                 f'dY4},_dY5={dY5},_s={s},_k={k},_v={v},_q={q}\n\n')
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    dY0 = np.linspace(lower, upper, points)
    Yaxis = []
    #Solve for each parameter value
    print("dY0_Bifurcation")
```

```
for j in tqdm(range(points)):
    Income = mapping(dY0[j], dY1, dY2, dY3, dY4, dY5, s, k, v, q, iterations)
    lastvalues = Income[-last:]
    Yaxis.extend(lastvalues)
return(Yaxis)

def y1bifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f'Initial_conditions_of_{dot}Y_1$bifurcation: lower={lower}, upper
                  ={upper}, points={points}, dY0={dY0}, dY2={dY2}, dY3={dY3}, dY4={
                  dY4}, dY5={dY5}, s={s}, k={k}, v={v}, q={q}\n\n')
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    dY1 = np.linspace(lower, upper, points)
    Yaxis = []
    #Solve for each parameter value
    print("dY1_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1[j], dY2, dY3, dY4, dY5, s, k, v, q, iterations)
        lastvalues = Income[-last:]
        Yaxis.extend(lastvalues)
    return(Yaxis)

def y2bifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f'Initial_conditions_of_{dot}Y_2$bifurcation: lower={lower}, upper
                  ={upper}, points={points}, dY0={dY0}, dY1={dY1}, dY3={dY3}, dY4={
                  dY4}, dY5={dY5}, s={s}, k={k}, v={v}, q={q}\n\n')
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    dY2 = np.linspace(lower, upper, points)
    Yaxis = []
```

```
#Solve for each parameter value
print("dY2_Bifurcation")
for j in tqdm(range(points)):
    Income = mapping(dY0, dY1, dY2[j], dY3, dY4, dY5, s, k, v, q, iterations)
    lastvalues = Income[-last:]
    Yaxis.extend(lastvalues)
return(Yaxis)

def y3bifurcation(lower, upper, points, dY0, dY1, dY2, dY4, dY5, s, k, v, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f'Initial_conditions_of_{dot}_Y_3_bifurcation: lower={lower}, upper={upper}, points={points}, dY0={dY0}, dY1={dY1}, dY2={dY2}, dY4={dY4}, dY5={dY5}, s={s}, k={k}, v={v}, q={q}\n\n')
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    dY3 = np.linspace(lower, upper, points)
    Yaxis = []
    #Solve for each parameter value
    print("dY3_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3[j], dY4, dY5, s, k, v, q, iterations)
        lastvalues = Income[-last:]
        Yaxis.extend(lastvalues)
    return(Yaxis)

def y4bifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY5, s, k, v, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f'Initial_conditions_of_Y_4_bifurcation: lower={lower}, upper={upper}, points={points}, dY0={dY0}, dY1={dY1}, dY2={dY2}, dY3={dY3}, dY5={dY5}, s={s}, k={k}, v={v}, q={q}\n\n')
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
```

```
dY4 = np.linspace(lower, upper, points)
Yaxis = []
#Solve for each parameter value
print("dY4_Bifurcation")
for j in tqdm(range(points)):
    Income = mapping(dY0, dY1, dY2, dY3, dY4[j], dY5, s, k, v, q, iterations)
    lastvalues = Income[-last:]
    Yaxis.extend(lastvalues)
return(Yaxis)

def y5bifurcation(lower, upper, points, dY0, dY1, dY2, dY3, dY4, s, k, v, q):
    #Print initial conditions of bifurcation to bifur_results.txt
    results.write(f"Initial_conditions_of_{Y}_bifurcation: lower={lower}, upper={upper}, points={points}, dY0={dY0}, dY1={dY1}, dY2={dY2}, dY3={dY3}, dY4={dY4}, s={s}, k={k}, v={v}, q={q}\n\n")
    #Plot set up
    last = 30
    iterations = 3000
    #Generate distribution of parameter
    dY5 = np.linspace(lower, upper, points)
    Yaxis = []
    #Solve for each parameter value
    print("dY5_Bifurcation")
    for j in tqdm(range(points)):
        Income = mapping(dY0, dY1, dY2, dY3, dY4, dY5[j], s, k, v, q, iterations)
        lastvalues = Income[-last:]
        Yaxis.extend(lastvalues)
    return(Yaxis)

#Saves results in a .txt file and appends further data to bottom of file
#Precision typically 0 indicating only whole number changes in growth are appreciable
changes in periodicity
results = open( './computational/metzlerian_growth/bifur-init_results.txt', 'a+')
Yaxis = y0bifurcation(-400, 400, 10000, 120, 110, 100, 105, 107, 0.6, 0.3, 500, 0.001)
lbifurana(-400, 400, 10000, Yaxis, -400, 1, 0)
rbifurana(-400, 400, 10000, Yaxis, 0, 2, 0)
```

lbifurana(-400, 400, 10000, Yaxis, 0, 2, 0)  
Yaxis = y1bifurcation(-400, 400, 10000, 100, 110, 100, 105, 107, 0.6, 0.3, 500, 0.001)  
lbifurana(-400, 400, 10000, Yaxis, -400, 1, 0)  
rbifurana(-400, 400, 10000, Yaxis, 100, 2, 0)  
lbifurana(-400, 400, 10000, Yaxis, 100, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, 400, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, 260, 1, 0)  
lbifurana(-400, 400, 10000, Yaxis, 260, 1, 0)  
Yaxis = y2bifurcation(-400, 400, 10000, 100, 120, 100, 105, 107, 0.6, 0.3, 500, 0.001)  
lbifurana(-400, 400, 10000, Yaxis, -400, 1, 0)  
rbifurana(-400, 400, 10000, Yaxis, 100, 2, 0)  
lbifurana(-400, 400, 10000, Yaxis, 100, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, 400, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, 260, 1, 0)  
lbifurana(-400, 400, 10000, Yaxis, 260, 1, 0)  
Yaxis = y3bifurcation(-400, 400, 10000, 100, 120, 110, 105, 107, 0.6, 0.3, 500, 0.001)  
lbifurana(-400, 400, 10000, Yaxis, -400, 1, 0)  
rbifurana(-400, 400, 10000, Yaxis, 200, 2, 0)  
lbifurana(-400, 400, 10000, Yaxis, 200, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, 400, 1, 0)  
Yaxis = y4bifurcation(-400, 400, 10000, 100, 120, 110, 100, 107, 0.6, 0.3, 500, 0.001)  
lbifurana(-400, 400, 10000, Yaxis, -400, 1, 0)  
rbifurana(-400, 400, 10000, Yaxis, -140, 2, 0)  
lbifurana(-400, 400, 10000, Yaxis, -140, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, -20, 2, 0)  
lbifurana(-400, 400, 10000, Yaxis, -20, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, 50, 1, 0)  
lbifurana(-400, 400, 10000, Yaxis, 50, 1, 0)  
rbifurana(-400, 400, 10000, Yaxis, 130, 2, 0)  
lbifurana(-400, 400, 10000, Yaxis, 130, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, 400, 1, 0)  
Yaxis = y5bifurcation(-400, 400, 10000, 100, 120, 110, 100, 105, 0.6, 0.3, 500, 0.001)  
lbifurana(-400, 400, 10000, Yaxis, -400, 1, 0)  
rbifurana(-400, 400, 10000, Yaxis, -0, 2, 0)  
lbifurana(-400, 400, 10000, Yaxis, -0, 2, 0)  
rbifurana(-400, 400, 10000, Yaxis, 120, 2, 0)

```
lbifurana(-400, 400, 10000, Yaxis, 120, 2, 0)
rbifurana(-400, 400, 10000, Yaxis, 300, 1, 0)
```

```
results.close()
```

## E.6 Results of Bifurcation Analysis

Initial conditions of s bifurcation: lower=0.1, upper=0.9, points=50000, dY0=100, dY1=120, dY2=110, dY3=100, dY4=105, dY5=107, k=0.3, v=500, q=0.001

Bifurcation Point found starting search from 0.53, left to right from 2 cycle. Bifurcation Point at 0.65. Long-run values at bifurcation: 131.0, 4.0, 132.0, 6.0, 133.0, 136.0, 135.0, 23.0, 31.0, 34.0, 46.0, 54.0, 67.0, 70.0, 76.0, 95.0, 98.0, 102.0, -19.0, -18.0, 110.0, 112.0, 114.0, -14.0, 118.0, -10.0, 120.0, 122.0, -1.0; Value Count = 29. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 0.53, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 0.65. Long-run values at bifurcation: 131.0, 4.0, 132.0, 6.0, 133.0, 136.0, 135.0, 23.0, 31.0, 34.0, 46.0, 54.0, 67.0, 70.0, 76.0, 95.0, 98.0, 102.0, -19.0, -18.0, 110.0, 112.0, 114.0, -14.0, 118.0, -10.0, 120.0, 122.0, -1.0; Value Count = 29. Analysis done rounding to 0 decimal places.

Initial conditions of k bifurcation: lower=0.1, upper=0.9, points=50000, dY0=100, dY1=120, dY2=110, dY3=100, dY4=105, dY5=107, s=0.6, v=500, q=0.001

Bifurcation Point found starting search from 0.1, left to right from 2 cycle. Bifurcation Point at 0.8483669673393468. Long-run values at bifurcation: -109.0, -99.0, -96.0, -91.0, 44.0, -83.0, -82.0, -84.0, -80.0, 51.0, -74.0, 55.0, -69.0, 61.0, -67.0, 67.0, -60.0, 73.0, -54.0, 75.0, 78.0, 87.0, -37.0, 91.0, 94.0, 96.0, 97.0, -30.0; Value Count = 28. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 0.9, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 0.8994559891197824. Long-run values at bifurcation: -207.0, 140.0, -75.0; Value Count = 3. Analysis done rounding to 0 decimal places.

Initial conditions of v bifurcation: lower=1, upper=2000, points=50000, dY0=100, dY1=120, dY2=110, dY3=100, dY4=105, dY5=107, s=0.6, k=0.3, q=0.001

Bifurcation Point found starting search from 1, left to right from 1 cycle. Bifurcation Point at 297.33768675373506. Long-run values at bifurcation: 101.0, 102.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 1000, right to left into 1 cycle. Bifurcation Point Found into 1 cycle. Bifurcation Point at 617.1041220824416. Long-run values at

bifurcation: 101.0, 102.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 525, left to right from 2 cycle. Bifurcation Point at 589.7572551451028. Long-run values at bifurcation: 74.0, 77.0, 79.0, 85.0, 90.0, 92.0, 94.0, 96.0, 97.0, 99.0, 100.0, 102.0, 104.0, 105.0, 106.0, 108.0, 109.0, 112.0, 113.0, 116.0, 119.0, 121.0, 123.0; Value Count = 23. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 525, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 472.17372347446945. Long-run values at bifurcation: 226.0, 83.0, -121.0; Value Count = 3. Analysis done rounding to 0 decimal places. Initial conditions of  $\dot{Y}_1$  bifurcation: lower=0, upper=0.002, points=50000, dY0=100, dY1=120, dY2=110, dY3=100, dY4=105, dY5=107, s=0.6, k=0.3, v=500

Bifurcation Point found starting search from 0.002, right to left into 1 cycle. Bifurcation Point Found into 1 cycle. Bifurcation Point at 0.0015997119942398846. Long-run values at bifurcation: 101.0, 102.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 0.001, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 0.0008898577971559431. Long-run values at bifurcation: 50.0, 51.0, 142.0; Value Count = 3. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 0.001, left to right from 2 cycle. Bifurcation Point at 0.0010441408828176563. Long-run values at bifurcation: 105.0, 107.0, 148.0, 21.0, 118.0, 22.0, 119.0, 59.0, 63.0; Value Count = 9. Analysis done rounding to 0 decimal places.

Initial conditions of  $\dot{Y}_0$  bifurcation: lower=-400, upper=400, points=10000, dY1=120, dY2=110, dY3=100, dY4=105, dY5=107, s=0.6, k=0.3, v=500, q=0.001

Bifurcation Point found starting search from -400, left to right from 1 cycle. Bifurcation Point at -267.986798679868. Long-run values at bifurcation: 145.0, 146.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 0, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at -180.37803780378036. Long-run values at bifurcation: 195.0, 196.0, 197.0, 87.0; Value Count = 4. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 0, left to right from 2 cycle. Bifurcation Point at 231.90319031903198. Long-run values at bifurcation: 104.0, -62.0, 78.0, -72.0; Value Count = 4. Analysis done rounding to 0 decimal places.

Initial conditions of  $\dot{Y}_1$  bifurcation: lower=-400, upper=400, points=10000, dY0=100, dY2=110, dY3=100, dY4=105, dY5=107, s=0.6, k=0.3, v=500, q=0.001

Bifurcation Point found starting search from -400, left to right from 1 cycle. Bifurcation Point at -99.00990099009897. Long-run values at bifurcation: 129.0, 139.0, 142.0, 153.0, 157.0, 167.0, 174.0, 185.0, 194.0, 202.0, 208.0, 210.0, 214.0, 87.0, 89.0, 218.0, 91.0, 220.0, 219.0, 92.0, 95.0, 96.0, 102.0, 106.0, 113.0, 115.0, 123.0; Value Count = 27. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 100, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at -48.204820482048206. Long-run values at bifurcation: 196.0, 197.0, 87.0; Value Count = 3. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 100, left to right from 2 cycle. Bifurcation Point at 220.54205420542053. Long-run values at bifurcation: 97.0, 67.0, 99.0, 101.0, 68.0, 73.0, 76.0, 77.0, 92.0, 80.0, 82.0, 83.0, 85.0, 86.0, 88.0, 89.0, 60.0, 93.0; Value Count = 18. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 400, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 313.2713271327133. Long-run values at bifurcation: 48.0; Value Count = 1. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 260, right to left into 1 cycle. Bifurcation Point Found into 1 cycle. Bifurcation Point at 243.10431043104313. Long-run values at bifurcation: 78.0, 79.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 260, left to right from 1 cycle. Bifurcation Point at 313.35133513351343. Long-run values at bifurcation: -61.0, -108.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Initial conditions of  $\dot{Y}_2$  bifurcation: lower=-400, upper=400, points=10000, dY0=100, dY1=120, dY3=100, dY4=105, dY5=107, s=0.6, k=0.3, v=500, q=0.001

Bifurcation Point found starting search from -400, left to right from 1 cycle. Bifurcation Point at -109.01090109010897. Long-run values at bifurcation: 129.0, 132.0, 139.0, 142.0, 148.0, 153.0, 157.0, 162.0, 167.0, 174.0, 177.0, 185.0, 194.0, 202.0, 210.0, 214.0, 87.0, 218.0, 219.0, 220.0, 92.0, 91.0, 95.0, 96.0, 106.0, 108.0, 113.0, 115.0; Value Count = 28. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 100, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at -58.20582058205821. Long-run values at bifurcation: 196.0, 197.0, 87.0; Value Count = 3. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 100, left to right from 2 cycle. Bifurcation Point at 210.54105410541058. Long-run values at bifurcation: 97.0, 99.0, 67.0, 101.0, 69.0, 73.0, 77.0, 92.0, 81.0, 84.0, 86.0, 87.0, 89.0, 59.0, 60.0; Value Count = 15. Analysis done

rounding to 0 decimal places.

Bifurcation Point found starting search from 400, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 303.27032703270334. Long-run values at bifurcation: 48.0; Value Count = 1. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 260, right to left into 1 cycle. Bifurcation Point Found into 1 cycle. Bifurcation Point at 233.10331033103319. Long-run values at bifurcation: 78.0, 79.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 260, left to right from 1 cycle. Bifurcation Point at 303.35033503350337. Long-run values at bifurcation: -61.0, -108.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Initial conditions of  $\dot{Y}_3$  bifurcation: lower=-400, upper=400, points=10000, dY0=100, dY1=120, dY2=110, dY4=105, dY5=107, s=0.6, k=0.3, v=500, q=0.001

Bifurcation Point found starting search from -400, left to right from 1 cycle. Bifurcation Point at -393.67936793679365. Long-run values at bifurcation: -120.0, -119.0, -112.0, -111.0, -104.0, -101.0, -100.0, -96.0, -95.0, -92.0, -90.0, -89.0, -217.0, -218.0, -214.0, -213.0, -210.0, -209.0, -203.0, -197.0, -194.0, -188.0, -177.0, -173.0, -165.0, -162.0, -146.0, -135.0; Value Count = 28. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 200, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 19.161916191619184. Long-run values at bifurcation: 104.0, -62.0, 78.0, -72.0; Value Count = 4. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 200, left to right from 2 cycle. Bifurcation Point at 270.86708670867085. Long-run values at bifurcation: 195.0, 196.0, 197.0, 86.0, 87.0; Value Count = 5. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 400, right to left into 1 cycle. Bifurcation Point Found into 1 cycle. Bifurcation Point at 324.2324232423242. Long-run values at bifurcation: 145.0, 146.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Initial conditions of  $Y_4$  bifurcation: lower=-400, upper=400, points=10000, dY0=100, dY1=120, dY2=110, dY3=100, dY5=107, s=0.6, k=0.3, v=500, q=0.001

Bifurcation Point found starting search from -400, left to right from 1 cycle. Bifurcation Point at -208.7008700870087. Long-run values at bifurcation: -102.0, 77.0, -355.0; Value Count = 3. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from -140, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at -144.05440544054403. Long-run values at bifurcation: -88.0, -252.0, 117.0; Value Count = 3. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from -140, left to right from 2 cycle. Bifurcation Point at -111.57115711571157. Long-run values at bifurcation: -104.0, -78.0, 62.0, 71.0; Value Count = 4. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from -20, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at -57.80578057805781. Long-run values at bifurcation: -128.0, 257.0, -126.0, -124.0, -123.0, -107.0, -102.0, -101.0, 53.0, 61.0, 62.0, 64.0, 69.0, 201.0, 74.0, 203.0, 206.0, 102.0, 103.0, 104.0, -146.0, -145.0, -143.0, 243.0, 244.0, 249.0, 254.0; Value Count = 27. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from -20, left to right from 2 cycle. Bifurcation Point at 2.8402840284028343. Long-run values at bifurcation: -63.0, 76.0, -74.0, 103.0; Value Count = 4. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 50, right to left into 1 cycle. Bifurcation Point Found into 1 cycle. Bifurcation Point at 19.72197219721977. Long-run values at bifurcation: 78.0, 79.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 50, left to right from 1 cycle. Bifurcation Point at 78.20782078207822. Long-run values at bifurcation: 78.0, 79.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 130, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 87.88878887888791. Long-run values at bifurcation: 100.0, 103.0, 74.0, 45.0, 80.0, 113.0, 85.0, 88.0, 60.0; Value Count = 9. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 130, left to right from 2 cycle. Bifurcation Point at 143.4943494349435. Long-run values at bifurcation: 195.0, 196.0, 197.0, 86.0, 87.0, 88.0; Value Count = 6. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 400, right to left into 1 cycle. Bifurcation Point Found into 1 cycle. Bifurcation Point at 158.05580558055806. Long-run values at bifurcation: 131.0, 139.0, 142.0, 152.0, 155.0, 167.0, 179.0, 180.0, 190.0, 192.0, 199.0, 201.0, 206.0, 207.0, 208.0, 210.0, 211.0, 94.0, 95.0, 96.0, 98.0, 101.0, 103.0, 108.0, 111.0, 117.0, 120.0, 127.0; Value Count = 28. Analysis done rounding to 0 decimal places.

Initial conditions of  $Y_5$  bifurcation: lower=-400, upper=400, points=10000, dY0=100, dY1=120, dY2=110, dY3=100, dY4=105, s=0.6, k=0.3, v=500, q=0.001

Bifurcation Point found starting search from -400, left to right from 1 cycle. Bifurcation Point at -104.05040504050402. Long-run values at bifurcation: -146.0, -145.0; Value Count = 2. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 0, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at -28.84288428842882. Long-run values at

bifurcation: 120.0, -87.0, -247.0; Value Count = 3. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 0, left to right from 2 cycle. Bifurcation Point at 27.242724272427267. Long-run values at bifurcation: 59.0; Value Count = 1. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 120, right to left into 2 cycle. Bifurcation Point Found into 2 cycle. Bifurcation Point at 71.16711671167121. Long-run values at bifurcation: -73.0, -62.0, 77.0, 103.0; Value Count = 4. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 120, left to right from 2 cycle. Bifurcation Point at 179.8979897989799. Long-run values at bifurcation: 196.0, 197.0, 87.0; Value Count = 3. Analysis done rounding to 0 decimal places.

Bifurcation Point found starting search from 300, right to left into 1 cycle. Bifurcation Point Found into 1 cycle. Bifurcation Point at 202.62026202620268. Long-run values at bifurcation: 129.0, 130.0, 139.0, 143.0, 152.0, 154.0, 157.0, 166.0, 179.0, 181.0, 188.0, 190.0, 198.0, 200.0, 203.0, 205.0, 208.0, 96.0, 97.0, 98.0, 99.0, 103.0, 105.0, 108.0, 111.0, 118.0, 121.0, 126.0; Value Count = 28. Analysis done rounding to 0 decimal places.

## E.7 Lyapunov Plot varying Parameters

```
import numpy as np
import math
import matplotlib.pyplot as plt
from tqdm import tqdm
from scipy.linalg import sqrtm

def growth(dYt1, dYt2, dYt3, dYt5, dYt6, s, k, v, q):
    dYt = (dYt1 / v) / ((dYt1 / v)**4 + q) - (dYt2 / v) / ((dYt2 / v)**4 + q) +
        ((k+1)/3) * ((1-s)*(dYt2-dYt5)+s*(dYt3-dYt6)) + (1-s)*dYt2 + s*dYt3
    return dYt

def mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q, iter):
    #Initialize vectors
    dY = np.append([dY0, dY1, dY2, dY3, dY4, dY5], np.zeros(iter-6))
    #Simulate
    for t in range(6, iter):
```

```
dY[t] = growth(dY[t-1], dY[t-2], dY[t-3], dY[t-5], dY[t-6], s, k, v, q,)

return dY

def partial1(dYt1, s, k, v, q):
    return -( ( 4*dYt1**4 ) / ( v**5 * ( q + ( ( dYt1**4 ) / v**4 ) )**2 ) ) + ( 1 / (
        v * ( q + ( ( dYt1**4 ) / v**4 ) ) ) )

def partial2(dYt2, s, k, v, q):
    return 1 + ( ( k + 1 ) * ( 1 - s ) ) / 3 - s + ( 4*dYt2**4 ) / ( v**5 * ( q + ( (
        dYt2**4 / v**4 ) ) )**2 ) - 1 / ( v * ( q + ( dYt2**4 / v**4 ) ) )

def jacobianmake(dYt1, dYt2, s, k, v, q):
    return np.array([[partial1(dYt1,s,k,v,q),partial2(dYt2,s,k,v,q),s+((k+1)/3)*s,0,((k
        +1)*(s-1))/3,-((k+1)*s)
        /3],[1,0,0,0,0,0],[0,1,0,0,0,0],[0,0,1,0,0,0],[0,0,0,1,0,0],[0,0,0,0,1,0]])

def lyapunov(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q, iter):
    dY = mapping(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q, iter)
    Jacobianlag = jacobianmake(dY[1], dY[0], s, k, v, q)
    vector = np.array([[1],[0],[0],[0],[0],[0]])
    stretch = 0
    for i in range(2, iter):
        Jacobian = jacobianmake(dY[i], dY[i-1], s, k, v, q)
        stretch += np.log(np.linalg.norm(np.matmul(Jacobian, vector)))
        vector = np.matmul(Jacobianlag, vector) / np.linalg.norm(np.matmul(Jacobian,
            vector))
        Jacobianlag = Jacobian
    lyexp = stretch / (iter-2)
    return lyexp

def slyplot(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, k, v, q, iter):
    #Plot set up
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    s = np.linspace(lower, upper, points)
    xaxis = s
    yaxis = np.ones(points)
    for j in tqdm(range(points)):
        yaxis[j] = lyapunov(dY0, dY1, dY2, dY3, dY4, dY5, s[j], k, v, q, iter)
```

```
ax.plot(xaxis, yaxis, ',k', alpha=0.25)
#Labelling
ax.minorticks_on()
ax.set_xlabel('$s$')
ax.set_ylabel('$\lambda$')

def klyplot(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, v, q, iter):
    #Plot set up
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    k = np.linspace(lower, upper, points)
    xaxis = k
    yaxis = np.ones(points)
    for j in tqdm(range(points)):
        yaxis[j] = lyapunov(dY0, dY1, dY2, dY3, dY4, dY5, s, k[j], v, q, iter)
        ax.plot(xaxis, yaxis, ',k', alpha=0.25)
        #Labelling
        ax.minorticks_on()
        ax.set_xlabel('$k$')
        ax.set_ylabel('$\lambda$')

def vlyplot(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, k, q, iter):
    #Plot set up
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    v = np.linspace(lower, upper, points)
    xaxis = v
    yaxis = np.ones(points)
    for j in tqdm(range(points)):
        yaxis[j] = lyapunov(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v[j], q, iter)
        ax.plot(xaxis, yaxis, ',k', alpha=0.25)
        #Labelling
        ax.minorticks_on()
        ax.set_xlabel('$v$')
```

```
    ax.set_ylabel('$\lambda$')

def qlyplot(lower, upper, points, dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, iter):
    #Plot set up
    fig = plt.figure(dpi=1200)
    ax = fig.add_subplot(111)
    #Generate distribution of parameter
    q = np.linspace(lower, upper, points)
    xaxis = q
    yaxis = np.ones(points)
    for j in tqdm(range(points)):
        yaxis[j] = lyapunov(dY0, dY1, dY2, dY3, dY4, dY5, s, k, v, q[j], iter)
    ax.plot(xaxis, yaxis, 'k', alpha=0.25)
    #Labelling
    ax.minorticks_on()
    plt.ticklabel_format(style='sci', axis='x', scilimits=(0,0))
    ax.set_xlabel('q')
    ax.set_ylabel('$\lambda$')

slyplot(0.1, 0.9, 20000, 100, 120, 110, 100, 105, 107, 0.3, 500, 0.001, 1000)
plt.savefig('./manuscript/figures/metzlerian_growth/slyplot.eps', dpi=2880)
klyplot(0.1, 0.9, 20000, 100, 120, 110, 100, 105, 107, 0.6, 500, 0.001, 1000)
plt.savefig('./manuscript/figures/metzlerian_growth/klyplot.eps', dpi=2880)
vlyplot(1, 2000, 20000, 100, 120, 110, 100, 105, 107, 0.6, 0.3, 0.001, 1000)
plt.savefig('./manuscript/figures/metzlerian_growth/vlyplot.eps', dpi=2880)
qlyplot(0, 0.002, 20000, 100, 120, 110, 100, 105, 107, 0.6, 0.3, 500, 1000)
plt.savefig('./manuscript/figures/metzlerian_growth/qlyplot.eps', dpi=2880)
```